EXPLOITING CONTACTS FOR INTERACTIVE CONTROL OF ANIMATED HUMAN CHARACTERS

A Thesis
Presented to
The Academic Faculty

by

Sumit Jain

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
August 2011

EXPLOITING CONTACTS FOR INTERACTIVE CONTROL OF ANIMATED HUMAN CHARACTERS

Approved by:

Professor C. Karen Liu, Advisor
College of Computing
Georgia Institute of Technology

Professor Irfan Essa
College of Computing
Georgia Institute of Technology

Professor Greg Turk
College of Computing
Georgia Institute of Technology

Professor Jun Ueda
School of Mechanical Engineering
Georgia Institute of Technology

Professor Victor B. Zordan
Department of Computer Science and Engineering
University of California Riverside

Date Approved: 28 June 2011

To my parents, Veena and Rakesh Kumar Jain,

for their eternal love and support.

ACKNOWLEDGEMENTS

I first wish to thank my parents, Veena and Rakesh, and my brother Amit for guiding me throughout my life and at the same time having confidence in me to support pursuit of the goals of my choice. Having such a family, that has always been by my side through the toughest periods of my life, is a blessing.

I want to thank my wife, Neha, who has been a constant source of encouragement for me through all these years that I have known her. Apart from the unconditional love she has given me, she has been really patient and sacrificing for supporting my hard work and making me believe that it was worth the effort. I am extremely lucky to have wonderful parents-in-law, Geeta and Vinay, who have been caring and supportive all these years.

I want to thank my friends Nipun Kwatra and Dhruv Mahajan who have been a great source of inspiration and fun since high school years. It has been a privilege to be able to interact with them regarding any technical or personal matters.

I want to thank the members of my dissertation committee: Irfan Essa, Greg Turk, Jun Ueda, Victor Zordan and my advisor Karen Liu, for their insightful suggestions and helping me look into my research work with a wider perspective.

Georgia Institute of Technology has given me the unique opportunity to interact with one of the best minds in the world. I want to thank the faculty members and my lab-mates and friends in the computer graphics group, with whom I have been interacting all these years: Professors Jarek Rossignac, Greg Turk, Irfan Essa and Karen Liu, and the students Yuting Ye, Chris Wojtan, Olga Symonova, Huamin Wang, Matt Flagg, Jason Williams, Brian Whited, Kihwan Kim, Topraj Gurung, Mark Luffel, Karthik Raveendran, Jie Tan, Sehoon Ha, Yunfei Bai and Kristin Siu.

Please excuse me if I missed mentioning anyone. The weekly "Geometry Group" meetings provided an excellent platform for formal and informal interaction with these people and helped shaping me into a better researcher as I was part of or witnessed the intriguing discussions on a wide range of topics related to computer graphics. I would especially want to thank Yuting Ye who started as a graduate student with Karen around the same time as myself. With numerous discussions we have had, she has been a great deal of help during my graduate career in overcoming many technical barriers and I owe my understanding of many technical concepts to her. It has been a pleasure to have had her around.

Finally, I wish to express my sincerest gratitude to my advisor Karen Liu, who has been the most instrumental in shaping me as a researcher. Along with her brilliance, it is her remarkable confidence and the never-say-die attitude that makes her such a great mentor. Her early insights into research problems and the ability to think far ahead has helped me carve out my research over the last six years. I feel immensely fortunate to have her as my advisor and am confident in saying that I would not have been as successful as I am today without her guidance. Thank you Karen!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

One of the common research goals in disciplines such as computer graphics and robotics is to understand the subtleties of human motion and develop tools for recreating natural and meaningful motion. Physical simulation of virtual human characters is a promising approach since it provides a testbed for developing and testing control strategies required to execute various human behaviors. Designing generic control algorithms for simulating a wide range of human activities, which can robustly adapt to varying physical environments, has remained a primary challenge.

This dissertation introduces methods for generic and robust control of virtual characters in an interactive physical environment. Our approach is to use the information of the physical contacts between the character and her environment in the control design. We leverage high-level knowledge of the kinematics goals and the interaction with the surroundings to develop active control strategies that robustly adapt to variations in the physical scene. For synthesizing intentional motion requiring long-term planning, we exploit properties of the physical model for creating efficient and robust controllers in an interactive framework. The control design leverages the reference motion capture data and the contact information with the environment for interactive long-term planning. Finally, we propose a compact *soft contact* model for handling contacts for rigid body virtual characters. This model aims at improving the robustness of existing control methods without adding any complexity to the control design and opens up possibilities for new control algorithms to synthesize agile human motion.

CHAPTER I

INTRODUCTION

Natural animal motion has fascinated scientists and engineers for centuries. Human motion is the most relevant and observable motion of all the animals. Yet, understanding the precise nature of human movements and complexities involved in modeling and controlling the human body has remained an unsolved problem. One of the common approaches of the researchers and scientists in the disciplines such as computer graphics and robotics is to observe human behavior to understand natural human control. An in-depth understanding of the subtleties of human behavior would give them the ability to develop tools for recreating human motion. Physical simulation of virtual human characters is a promising approach as it provides a platform for developing and testing control strategies required to execute various human behaviors. Such a framework enables us to create novel human motion that alleviates the need to capturing real human motion in potentially dangerous situations such as stunt shots in movies or fighting scenarios in computer games.

Early research in physical simulation of characters focused on designing control methods for synthesizing specific human activities. However, the quest of synthesizing more complex behaviors demands robust and general control techniques that can be adapted to a variety of situations. In addition to the control forces generated internally by the human body, physical interaction with the surroundings plays a crucial role in performing any activity. The contacts with the environment provide kinematic constraints and external forces to affect the motion thereby being an important aspect of control. In this dissertation, we focus on exploring *generic control algorithms* that exploit the physical contacts between the character and the environment in the

control design. We further investigate the impact of *soft contacts* on the robustness of various control algorithms designed for physically simulated virtual characters.

There are two important aspects for animation of virtual characters: First, physical modeling and second, simulation and control. Physical modeling includes physical descriptions of the character such as the mass and inertia properties along with how the anatomy of the human body is abstracted into a practical physical structure. The human body consists of 206 distinct bones and an extremely complex neuromuscular system with an average of 300 trillion cells. Modeling the entire human body is impractical due to limited scientific tools and lack of complete knowledge of the human body. Figure 1 illustrates the complex musculoskeletal structure of the human body. In practice, vast simplifications are made in modeling the human body character that are practically suited for building control mechanisms mimicking some aspects of the human motion. For physics-based simulation and control of virtual characters, articulated rigid body models are extremely popular. Figure 2 illustrates the abstraction of the complex human body into an articulated rigid body system connected through different types of rotational joints. Robots that are built to emulate the human motion are also commonly constructed as rigid bodies connected through joints that are powered by motors. These simplified models have continued to provide a foundation for building advanced control algorithms for physical characters in different environments. However, it is necessary to bridge the gap between these simple models and the human anatomy in hope to successfully capture the nuances of natural human motion. For example, the effects of the skin that covers the bone and interacts with the environment through contacts, and the complex muscle structure that provides necessary forces to the human body are hard to replicate using the simple models. It is not immediately clear whether there are some enhancements to these simple models that can help improve the control problem drastically. In addition, it is necessary to avoid adding complexities that could make the modeling and control impractical.

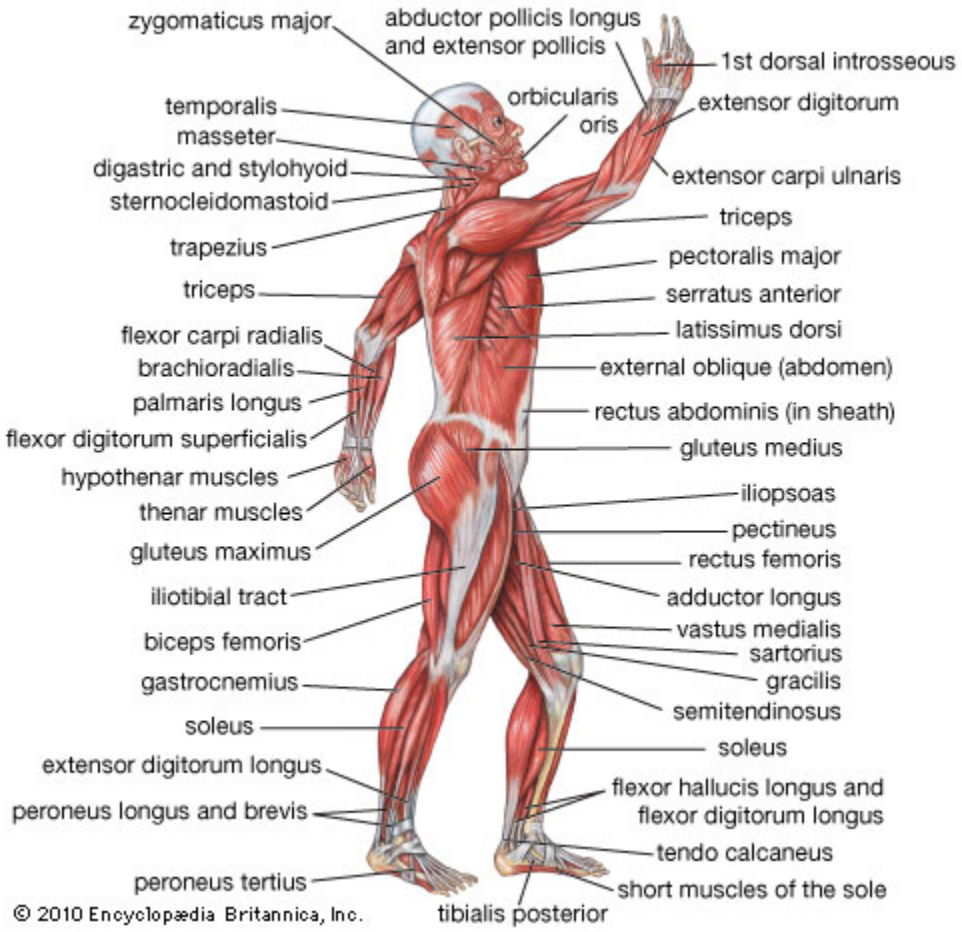


Figure 1: Illustration of the human musculoskeletal system. Photograph retrieved on June 29, 2011 from website: `http://www.britannica.com`

Extensive research in the animation and robotics disciplines has advanced our understanding and ability to controlling the physical models such as an articulated rigid body system. However, apart from the physical model of the character, the success of any control algorithm in the simulated or the real world is closely tied to handling the interaction between the character and her environment. Most of the human activities involve frequent interactions with the physical surroundings such as taking support from the ground while walking or running. Contacts with the environment provide kinematic constraints and external forces to counteract under-actuation (the inability to directly control the position and orientation of the entire body). Rather than passively computing the contact forces i.e. computing the contact

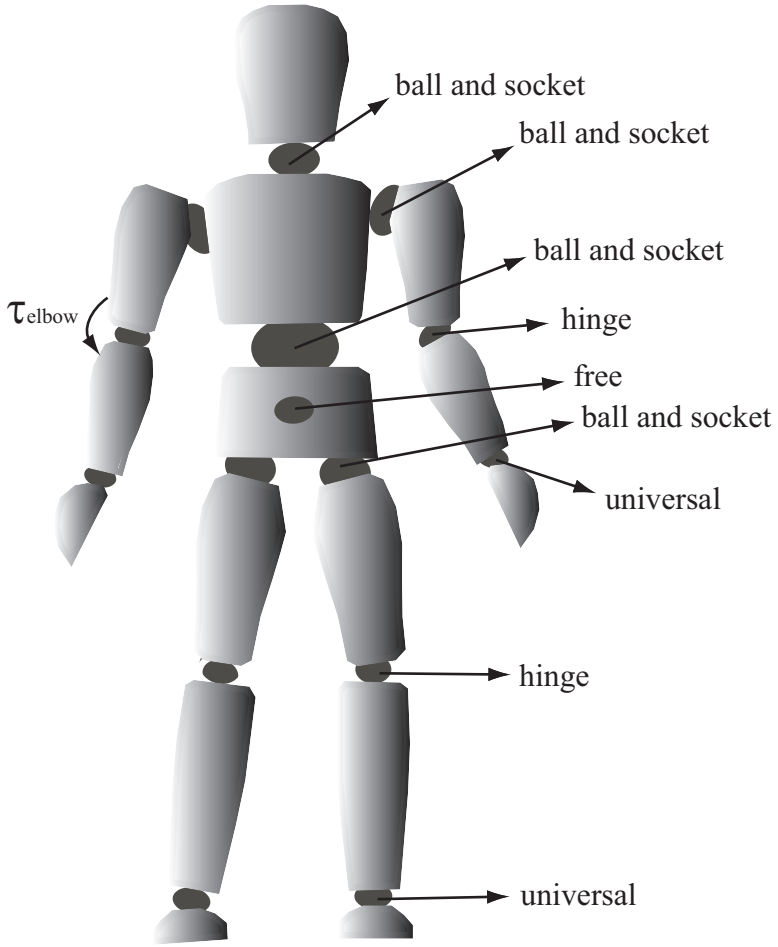


Figure 2: Illustration of an articulated rigid body structure. The bones are modeled as rigid bodies connected through *hinge*, *universal* or *ball and socket* joints with 1, 2 and 3 rotation DOFs respectively. The *free* joint connects the root of the structure to the world frame with 3 translation and 3 rotation DOFs. The muscle forces are abstracted as joint torques at all the joints except the *free* joint. As an example, the torque applied at the elbow is denoted by τ_{elbow}.

forces as a reaction to the control forces, the control algorithms may use the contact information to simultaneously compute the control and contact forces. This approach makes the control algorithms "aware" of the contacts, thereby directly incorporating their effect on the control forces during the computation.

In this dissertation, our goal is to develop generic and robust control algorithms for interactive physical environments that contribute to a better understanding of human control. The important challenges in controller design for interactive environments are

robustness and *efficiency* in the control computation. Traditional controllers involved significant effort in tuning of control parameters for synthesizing specific motions. These controllers being highly efficient worked only for a narrow range of conditions. Our approach is to use the information of the physical contacts between the character and her environment in the control design. This allows us to build robust control algorithms for interactive environments. We leverage high-level knowledge of the kinematics goals and the interaction with the surroundings to develop active control strategies that robustly adapt to variations in the physical scene. For synthesizing intentional motion requiring long-term planning, we exploit properties of the physical model for creating an efficient and a robust controller. The control design leverages the reference motion capture data and the contact information with the environment for interactive long-term planning. Finally, we propose a compact *soft contact* model for handling contacts for rigid body virtual characters. The soft contact formulation aids the existing control algorithms in performing more robustly while opening up a new dimension that the future control algorithms can leverage to synthesize agile human motion.

1.1 Animation of virtual characters

Animation implies specifying trajectories in time using analytical functions or discrete samples. Only a small subset of all the possible trajectories resembles natural or believable motion. Synthesizing meaningful animation is hard due to several factors. First, high dimensionality of the animation domain increases the complexity of motion synthesis. Second, several phenomenon are governed by the laws of nature that are encoded by complex equations of motion. Third, domain knowledge and artistic skills may be required to synthesize some motions as ensuring physical correctness may not lead to natural looking motion.

Physics-based animation of natural phenomenon such as water is usually associated with high dimensionality and complex physical equations of motion. Physics-based animation of virtual characters poses other important challenges as well. Simulation of characters represented as articulated rigid bodies is governed by physics as well as the "choice" of the actuator forces representing muscle and tendon forces that the character can apply to affect her motion. These internal forces add more degrees of freedom (DOFs) to the animation that are not entirely constrained or governed by any physical equations. This makes the animation system under-constrained implying that there are multiple solutions that satisfy some given constraints. However, for character animation, only a few such solutions may be desirable since all the possible motions may not look natural or believable. This qualitative measure for the synthesized motion makes the problem of synthesizing natural human-like motion even harder. Finally, the human character is under-actuated i.e. the character cannot generate forces that can control the global position and orientation of the center of mass (COM) of the entire body. This poses an important challenge for the control of full body human motion since the COM can only be controlled indirectly through contacts with the environment. In addition, the forces at these contact points do zero work and can only push apart.

The redundancy in human control and the observation and knowledge of everyday human motion can help to counter some of the challenges posed by the physics-based methods. Data-driven techniques use captured motions of a real human subject to acquire naturalness of human motion. High fidelity captured motion can be very expressive and is hard to synthesize using other techniques. Data-driven methods synthesize motions using a database of captured motion sequences. The synthesized motions are always similar to the ones in the database. Therefore, to synthesize a wide range of behaviors, it is necessary to have a large database of motions that capture the desired behaviors and their variations. This makes data-driven techniques limited to

the available motion sequences rendering them unsuitable to producing novel motion sequences in different environments.

Despite many shortcomings of the data-driven techniques, captured motion simplify some of the challenges in physics-based motion synthesis. Most contemporary animation methods that simulate the character's motion in a physically simulated environment employ a combination of these techniques. In the next section, we discuss various approaches to character animation using physics-based techniques and the challenges posed by these methods.

1.2 Physically-based character animation

The ability to automatically synthesize responses to the changes in the interactive environment makes physics-based controllers a promising avenue to pursue for animation of virtual characters. The earliest work in physics-based controllers for animating full body human motion such as walking, running, jumping and sitting [38, 37, 36, 28] focused on computing the forces by manually tuning the control parameters required to follow motion pre-defined trajectories for every joint. However, this methodology was limited to synthesizing behaviors specialized for a given physical scene and the character thus making it hard to adapt to environments and characters with different physical descriptions. Recent advances in controller design [105, 22] have resulted in more robust controllers that adapt to changes in the physical properties of the character or the environment. The control force computation consists of feedback error correction not only for the joint angle trajectories but also for global orientation of the character required for maintaining balance. However, the feedback gains for error correction need to be manually determined. The joint trajectories are related to the accelerations (hence the forces) by integrating the accelerations twice resulting in cumbersome and non-intuitive process of determining gains for computing forces.

Physics-based controllers based on optimal feedback control [24, 69, 103] leverage

the captured motion data and the contact configuration to compute the optimal gains necessary to perform the given reference motion. This approach offers two advantages: First, the synthesized motion is similar to the reference motion thereby preserving the naturalness of the human motion. Second, the control parameters are computed using an offline optimal control calculation performed on the entire motion sequence that liberates the user from computing the control parameters manually. The controller is able to adjust to small perturbations or changes in the environment. However, in the scenarios that require a change in the strategy to counter any changes to the physical setup, the method fails since the reference motion trajectory cannot be changed online during the simulation and the control parameters need to be recomputed.

In addition to control, virtual modeling of the character is an important aspect of character animation. For physics-based control of virtual human characters, the character is commonly represented as an articulated rigid body system that abstracts away the complex musculoskeletal structure of the human body with a tree structure of rigid bodies representing the bones that are connected through joints with rotation DOFs (Figure 2). Each joint is equipped with an actuator that can produce a torque modeling the muscle and tendon forces. The articulated rigid body model is effective in capturing the key aspects of the human behavior expressed in full body motion. However, there remains a huge gap in complexity between the articulated model and anatomical model that limits the synthesis of natural human motion. Many researchers have developed separate advanced models such as biomechanical models for neuromuscular control of the upper body and the neck [52, 51], hand models for detailed manipulation control [9, 86] and muscle and skin deformation models [70]. Though these methods drastically improve the aesthetics of the human model along with detailed local control, they do not target the improvement in full body control of the character. These advanced models tremendously increase the number of control parameters making the control formulation even harder. Therefore, the

added complexity and the inability to improve the robustness of the full body control problem limits the choice of these advanced models in controlling an under-actuated character.

1.3 Thesis contributions

Physics-based control for virtual characters offers a generic platform to synthesize novel motion sequences with varying physical properties of both the character and the environment. However, control design remains a tedious task with skilled engineers involved in the fine tuning of the control parameters to deliver robust controllers for dynamic virtual characters. As a result, there are a few questions that arise regarding the design of controllers: First, how can we leverage high-level domain knowledge in the design of complex control strategies such that it becomes easier to adapt the control to different physical configurations? Second, how can the control design leverage the physical properties of the character and her surroundings for robust and efficient control? Third, what is the most effective enhancement to the design of the physical model that can largely improve full body control with minimal complexity added to existing control algorithms?

In this thesis, we attempt to answer these questions by developing control methods that are generic to be adapted to different physical properties of the character and her environment. We focus on interactive methods that exploit the physical contact configuration of the character with her surroundings. In addition, we propose an enhancement to the articulated rigid body model that augments the character with deformable bodies at the sites of contact. This model aims at improving the robustness of existing control methods without adding any complexity to the control design. The contributions of this thesis are summarized as follows:

Optimization-based interactive motion synthesis. We depart from the existing approach of controlling virtual characters in an interactive setting using physics-based controllers, which compute forces to be applied to the character. We explore an alternative framework in Chapter 3 for active character simulation that formulates motion synthesis as a sequence of non-linear optimization problems. In this framework, the controller takes the state of the character and her contact configuration as input, and directs the motion by providing kinematic goals to the optimization.

Long-term control planning in modal space. We present a novel control algorithm for simulating full body motion of a character performing a given reference motion and its variations (Chapter 4). The input motion provides the algorithm with the reference joint angle trajectories and the contact configuration. Our algorithm computes the control parameters by solving a long-term plan at every time step thereby allowing a wide range of variations to the reference motion during online simulation and does not have dependency on any offline precomputation. Our framework exploits the natural frequencies of the physical system describing the character in formulating an efficient control problem in the dynamically decoupled modal coordinates.

Soft contacts for robust control. We hypothesize that modeling the virtual character as a deformable body at the site of contacts is an effective step towards synthesizing accurate and natural human-environment interaction with small added complexity. In Chapter 5, we introduce a compact representation for an articulated character with deformable meshes and develop a practical system to simulate two-way coupling between rigid and deformable bodies in a robust and efficient manner. We demonstrate the advantages of this approach by conducting experiments comparing our results with motion generated by existing control algorithms including biped locomotion.

CHAPTER II

BACKGROUND

In the context of physics-based animation, a human character is commonly represented as an articulated rigid body structure (Figure 2), which is a multi-body system of rigid bodies connected through joints. In this chapter, we derive the equations of motion for such a system of connected rigid bodies. We start by revisiting the dynamics for a single rigid body defined by the Newton-Euler equations in Section 2.1 and derive the Lagrange's equations of motion in the *generalized coordinates* in Section 2.2. We then derive the Lagrangian dynamics of an articulated rigid body in Section 2.3 by defining the set of generalized coordinates and extending the Lagrangian equations of a rigid body.

Generalized coordinates are a set of coordinates used to describe the configuration of a system relative to some reference configuration. Dynamics formulation in generalized coordinates has the advantage over the Cartesian coordinates since the equations of motion are parametrized by much fewer coordinates, often representing the independent degrees-of-freedom of the system, as opposed to the maximal representation with explicit constraints to enforce connectivity of the rigid bodies. This reduction in the parameters is often convenient for the formulation of control algorithms and simulation frameworks. Many physics simulators such as Open Dynamics Engine [5], PhysX [6], Havok [3] or Bullet Physics [2] use maximal representation in Cartesian coordinates rather than generalized coordinates. Since the parametrization is different for Cartesian and generalized coordinates, any quantity such as the position, the velocity or the force has different corresponding representations. Therefore,

to use the third-party physics simulators on top of the formulation in the generalized coordinates, it is important to correctly transform the quantities from one space to the other. In Section 2.4, we derive the transformations that convert quantities between the Cartesian and the generalized space.

2.1 Rigid body dynamics: Newton-Euler equations

This section summarizes the Newton-Euler equations of a single rigid body. We refer the reader to the SIGGRAPH course notes on Physically-Based Modeling by Witkin and Baraff for a comprehensive understanding of rigid body dynamics [93, 94].

Consider a rigid body whose mass, position of the center of mass (COM), rotation matrix corresponding to the orientation, linear velocity of the COM, and angular velocity are denoted by m, $\mathbf{x}$, R, $\mathbf{v}$, and $\boldsymbol{\omega}$ respectively. The linear momentum $\mathbf{p}$ is defined as $\mathbf{p} = m\mathbf{v}$. The angular momentum $\mathbf{L}$ is defined as $\mathbf{L} = I_c\boldsymbol{\omega}$, where I_c is the inertia tensor of the rigid body defined about the COM (see [93, 94]). Note that the inertia tensor can be written as $I_c = RI_0R^T$, where I_0 is the constant inertia tensor defined at zero rotation. The angular velocity in the skew-symmetric form is related to the rotation matrix R as $[\boldsymbol{\omega}] = \dot{R}R^T$. Also $[\boldsymbol{\omega}]^T = -[\boldsymbol{\omega}]$. The notation $[\mathbf{a}]\mathbf{b}$ denotes the cross product $\mathbf{a} \times \mathbf{b}$ with $[\mathbf{a}]$ being the skew-symmetric matrix corresponding to the vector $\mathbf{a}$:

$$[\mathbf{a}] = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \tag{1}$$

Therefore the following identities hold: $[\mathbf{a}]\mathbf{b} = -[\mathbf{b}]\mathbf{a}$ and $[\mathbf{a}]^T = -[\mathbf{a}]$.

Now, the dynamics of a rigid body can be written as $\mathbf{f} = \dot{\mathbf{p}}$, $\boldsymbol{\tau} = \dot{\mathbf{L}}$. Evaluating these time derivatives, we arrive at the Newton-Euler equations:

$$\begin{pmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & I_c \end{pmatrix} \begin{pmatrix} \dot{\mathbf{v}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\omega} \times I_c\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix} \tag{2}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix.

2.2 Rigid body dynamics: Lagrange's equations

We begin with giving a brief review of the Lagrangian dynamics for any general physical system. We apply these principles and derive the equations of motion for a rigid body.

2.2.1 Review of Lagrangian dynamics

Consider a physical system described by a vector of generalized coordinates $\mathbf{q} = (q_1, \ldots, q_n)^T$. Now, the position of any point $\mathbf{r}$ of the system, in the Cartesian space at time t, can be written as:

$$\mathbf{r} = \mathbf{r}(q_1, \ldots, \ldots, q_n, t) \tag{3}$$

The virtual displacement $\delta\mathbf{r}$ refers to an infinitesimal change in the system coordinates while the time is held constant, such that the constraints of the system remain satisfied. Equivalently, the virtual displacement $\delta\mathbf{r}$ is a tangent vector to the constraint manifold at a fixed time and can be written as:

$$\delta\mathbf{r} = \sum_j \frac{\partial \mathbf{r}}{\partial q_j} \delta q_j \tag{4}$$

We can write virtual work of force $\mathbf{f}$ acting on particle $\mathbf{r}$ as

$$\mathbf{f}.\delta\mathbf{r} = \mathbf{f}.\sum_j \frac{\partial \mathbf{r}}{\partial q_j} \delta q_j \equiv \sum_j Q_j \delta q_j = \mathbf{Q}.\delta\mathbf{q} \tag{5}$$

where $Q_j = \left(\frac{\partial \mathbf{r}}{\partial q_j}\right)^T \mathbf{f}$ is defined as the component of the *generalized force* associated with coordinate q_j. In the vector form, $\mathbf{Q}$ is the generalized force corresponding to the Cartesian force $\mathbf{f}$ with the relation $\mathbf{Q} = J^T\mathbf{f}$, where J is the Jacobian matrix with the j^{th} column defined as $\frac{\partial \mathbf{r}}{\partial q_j}$.

From the D'Alembert's principle and the definition of the generalized force in Equation 5, the *Lagrange's equations of motion* for each generalized coordinate q_j can be derived and written as:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} - Q_j = 0 \tag{6}$$

where T is the kinetic energy of the system. We write the Lagrange's equations in the vector form as:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} - \mathbf{Q} = 0 \tag{7}$$

The kinetic energy of a physical system can always be represented as $T = \frac{1}{2}\dot{\mathbf{q}}^T M(\mathbf{q})\dot{\mathbf{q}}$, where $M(\mathbf{q})$ is the *Mass matrix* that depends non-linearly on the configuration $\mathbf{q}$. Substituting the expression for T in Equation 7, the equations of motion in generalized coordinates can be written as:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q} \tag{8}$$

where $C(\mathbf{q}, \dot{\mathbf{q}})$ denotes the Coriolis and the centrifugal term that is quadratic in $\dot{\mathbf{q}}$.

2.2.2 Rigid body dynamics in generalized coordinates

The Newton-Euler equations for a rigid body in Equation 2 are defined in terms of the velocities instead of position and orientation. We now derive the equations in generalized coordinates $\mathbf{q}$ that define the position and orientation. The first three coordinates are the same as the position of COM. The next three represent the rotation of the rigid body such as an Exponential map or three Euler angles (or four coordinates can be used for a quaternion).

We start by computing the kinetic energy of the rigid body by summing up the kinetic energies of infinitesimal mass points:

$$\begin{aligned} T &= \sum_i T_i = \sum_i \frac{1}{2}\mu \dot{\mathbf{r}}_i^T \dot{\mathbf{r}}_i = \sum_i \frac{1}{2}\mu (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_i')^T (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_i') \\ &= \sum_i \frac{1}{2}\mu (\mathbf{v}^T\mathbf{v} + \mathbf{v}^T[\boldsymbol{\omega}]\mathbf{r}_i' + \mathbf{r}_i'^T[\boldsymbol{\omega}]^T\mathbf{v} + \mathbf{r}_i'^T[\boldsymbol{\omega}]^T[\boldsymbol{\omega}]\mathbf{r}_i') \end{aligned} \tag{9}$$

where μ is the mass of the infinitesimal mass point, $\mathbf{r}_i$ is its position in the space and $\mathbf{r}_i'$ is the vector from the COM of the rigid body to the mass point. Because $\sum_i \mu \mathbf{r}_i' = \mathbf{0}$ (property of the COM), the second term and the third term in Equation 9 vanish.

Using the identity $[\boldsymbol{\omega}]\mathbf{r}'_i = -[\mathbf{r}'_i]\boldsymbol{\omega}$, we can rewrite Equation 9 as:

$$\begin{aligned} T &= \frac{1}{2}m\mathbf{v}^T\mathbf{v} + \frac{1}{2}\boldsymbol{\omega}^T\left(\sum_i -\mu[\mathbf{r}'_i][\mathbf{r}'_i]\right)\boldsymbol{\omega} \\ &= \frac{1}{2}m\mathbf{v}^T\mathbf{v} + \frac{1}{2}\boldsymbol{\omega}^T I_c\boldsymbol{\omega} \end{aligned} \tag{10}$$

Note that we used the definition of the Inertia tensor I_c = The kinetic energy of a rigid body can be written in its vector form:

$$T = \frac{1}{2}(\mathbf{v}^T \;\; \boldsymbol{\omega}^T)\begin{pmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & I_c \end{pmatrix}\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix} \equiv \frac{1}{2}\mathbf{V}^T M_c\mathbf{V} \tag{11}$$

where $\mathbf{V} = (\mathbf{v}^T, \boldsymbol{\omega}^T)^T$, $M_c = \text{blockdiag}(m\mathbf{I}_3, I_c)$. We now relate the velocities in the Cartesian space $\mathbf{V}$ to the generalized velocities $\dot{\mathbf{q}}$. Let $\mathbf{x}(\mathbf{q})$ and $R(\mathbf{q})$ represent the position of the COM and the rotation matrix of the rigid body. The linear velocity of the COM is computed as:

$$\mathbf{v} = \dot{\mathbf{x}}(\mathbf{q}) = \frac{\partial\mathbf{x}}{\partial\mathbf{q}}\dot{\mathbf{q}} \equiv J_v\dot{\mathbf{q}} \tag{12}$$

The angular velocity is computed as:

$$\begin{aligned} [\boldsymbol{\omega}] &= \dot{R}(\mathbf{q})R^T(\mathbf{q}) \\ &= \sum_j \frac{\partial R}{\partial q_j}R^T\dot{q}_j \equiv \sum_j [\mathbf{j}_j]\dot{q}_j \end{aligned} \tag{13}$$

$\frac{\partial R}{\partial q_j}R^T$ is always a skew-symmetric matrix that we represent as $[\mathbf{j}_j]$ (skew-symmetric form of the vector $\mathbf{j}_j$). $\boldsymbol{\omega}$ can be now be represented in the vector form as:

$$\boldsymbol{\omega} = J_\omega\dot{\mathbf{q}} \tag{14}$$

where $\mathbf{j}_j$ is the j^{th} column of the matrix J_ω.

Using Equation 12 and Equation 14, we can write:

$$\mathbf{V} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix}\dot{\mathbf{q}} \equiv J(\mathbf{q})\dot{\mathbf{q}} \tag{15}$$

Substituting the above in Equation 11, we get:

$$T = \frac{1}{2}\dot{\mathbf{q}}^T J^T M_c J \dot{\mathbf{q}} \tag{16}$$

Using the recipe of Lagrangian dynamics in Equation 6, we first compute $\frac{\partial T}{\partial \dot{q}_j}$ as:

$$\begin{aligned} \frac{\partial T}{\partial \dot{q}_j} &= \frac{1}{2}\dot{\mathbf{q}}^T J^T M_c (J)_j + \frac{1}{2}(J)_j^T M_c J \dot{\mathbf{q}} \\ &= (J)_j^T M_c J \dot{\mathbf{q}} \end{aligned} \tag{17}$$

where the notation $(A)_j$ denotes the j^{th} column of the matrix A. The term $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right)$ is computed as:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) = (J)_j^T M_c J \ddot{\mathbf{q}} + (J)_j^T M_c \dot{J} \dot{\mathbf{q}} + (J)_j^T \dot{M}_c J \dot{\mathbf{q}} + (\dot{J})_j^T M_c J \dot{\mathbf{q}} \tag{18}$$

Now we evaluate the term $\frac{\partial T}{\partial q_j}$:

$$\begin{aligned} \frac{\partial T}{\partial q_j} &= \frac{1}{2}\dot{\mathbf{q}}^T J^T M_c \frac{\partial J}{\partial q_j}\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T J^T \frac{\partial M_c}{\partial q_j} J \dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T \frac{\partial J^T}{\partial q_j} M_c J \dot{\mathbf{q}} \\ &= \dot{\mathbf{q}}^T \frac{\partial J^T}{\partial q_j} M_c J \dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T J^T \frac{\partial M_c}{\partial q_j} J \dot{\mathbf{q}} \end{aligned} \tag{19}$$

Using the above equations, we write:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} &= (J)_j^T M_c J \ddot{\mathbf{q}} + (J)_j^T M_c \dot{J} \dot{\mathbf{q}} + (J)_j^T \dot{M}_c J \dot{\mathbf{q}} - \frac{1}{2}\dot{\mathbf{q}}^T J^T \frac{\partial M_c}{\partial q_j} J \dot{\mathbf{q}} \\ &\quad + \left((\dot{J})_j^T M_c J \dot{\mathbf{q}} - \left(\frac{\partial J}{\partial q_j}\dot{\mathbf{q}}\right)^T M_c J \dot{\mathbf{q}} \right) \end{aligned} \tag{20}$$

The second term in the above equation involves the computation of $\dot{J}$ that can be computed as $\sum_k \frac{\partial J}{\partial q_k}\dot{q}_k$. We now simplify the third, fourth and the fifth terms one by one. Let us start with the third term:

$$\begin{aligned} (J)_j^T \dot{M}_c J \dot{\mathbf{q}} &= (J_\omega)_j^T \dot{I}_c J_\omega \dot{\mathbf{q}} \ \text{(The linear term in } M_c \text{ is constant: see Equation 11)} \\ &= \mathbf{j}_j^T \dot{I}_c \boldsymbol{\omega} \ \ (\mathbf{j}_j \text{ represents the } j^{th} \text{ column of } J_\omega \text{: see Equation 13)} \\ \text{term 3} &= \mathbf{j}_j^T [\boldsymbol{\omega}] I_c \boldsymbol{\omega} \ \ (\text{Since } \dot{I}_c = (R\dot{I_0}R^T) = [\boldsymbol{\omega}] I_c) \end{aligned} \tag{21}$$

The fourth term in Equation 20 can be simplified as:

$$\begin{aligned}
\frac{1}{2}\dot{\mathbf{q}}^T J^T \frac{\partial M_c}{\partial q_j} J\dot{\mathbf{q}} &= \frac{1}{2}(J_\omega \dot{\mathbf{q}})^T \frac{\partial I_c}{\partial q_j} J_\omega \dot{\mathbf{q}} \\
&= \frac{1}{2}\boldsymbol{\omega}^T \left(\frac{\partial R}{\partial q_j} I_0 R^T + R I_0 \frac{\partial R^T}{\partial q_j} \right) \boldsymbol{\omega} = \boldsymbol{\omega}^T \left(\frac{\partial R}{\partial q_j} I_0 R^T \right) \boldsymbol{\omega} \\
&= \boldsymbol{\omega}^T \left(\frac{\partial R}{\partial q_j} R^T I_c \right) \boldsymbol{\omega} \\
&= \boldsymbol{\omega}^T [\mathbf{j}_j] I_c \boldsymbol{\omega} \quad \text{(From Equation 13)} \\
\text{term 4} &= -\mathbf{j}_j^T [\boldsymbol{\omega}] I_c \boldsymbol{\omega} \quad \text{(Using the identity } \mathbf{a}.(\mathbf{b}\times\mathbf{c}) = -\mathbf{b}.(\mathbf{a}\times\mathbf{c})) \qquad (22)
\end{aligned}$$

For simplifying the fifth term in Equation 20, we explicitly express it using the linear and angular components:

$$\begin{aligned}
\text{term 5} &= \begin{pmatrix} (\dot{J_v})_j{}^T & (\dot{J_\omega})_j{}^T \end{pmatrix} \begin{pmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & I_c \end{pmatrix} \begin{pmatrix} J_v \dot{\mathbf{q}} \\ J_\omega \dot{\mathbf{q}} \end{pmatrix} \\
&\quad - \begin{pmatrix} \left(\frac{\partial J_v}{\partial q_j}\dot{\mathbf{q}}\right)^T & \left(\frac{\partial J_\omega}{\partial q_j}\dot{\mathbf{q}}\right)^T \end{pmatrix} \begin{pmatrix} m\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & I_c \end{pmatrix} \begin{pmatrix} J_v \dot{\mathbf{q}} \\ J_\omega \dot{\mathbf{q}} \end{pmatrix} \qquad (23)
\end{aligned}$$

The linear term can be extracted and simplified as:

$$\begin{aligned}
m\left((\dot{J_v})_j - \left(\frac{\partial J_v}{\partial q_j}\dot{\mathbf{q}}\right)\right)^T J_v \dot{\mathbf{q}} &= m\left(\sum_k \frac{\partial (J_v)_j}{\partial q_k}\dot{q}_k - \sum_k \frac{\partial (J_v)_k}{\partial q_j}\dot{q}_k\right)^T J_v \dot{\mathbf{q}} \\
&= m\left(\sum_k \frac{\partial^2 \mathbf{x}}{\partial q_j \partial q_k}\dot{q}_k - \sum_k \frac{\partial^2 \mathbf{x}}{\partial q_k \partial q_j}\dot{q}_k\right)^T J_v \dot{\mathbf{q}} \\
\text{term 5 (linear)} &= 0 \qquad (24)
\end{aligned}$$

The above derivation uses the property of the Jacobian of the linear velocity $(J_v)_j = \frac{\partial \mathbf{x}}{\partial q_j}\ \forall j$ (See Equation 12).

We now extract and simplify the angular term in Equation 23 as:

$$\begin{aligned}
\left((\dot{J_\omega})_j - \left(\frac{\partial J_\omega}{\partial q_j}\dot{\mathbf{q}}\right)\right)^T I_c J_\omega \dot{\mathbf{q}} &= \left(\sum_k \frac{\partial \mathbf{j}_j}{\partial q_k}\dot{q}_k - \sum_k \frac{\partial \mathbf{j}_k}{\partial q_j}\dot{q}_k\right)^T I_c \boldsymbol{\omega} \\
&= \left(\sum_k \left(\frac{\partial \mathbf{j}_j}{\partial q_k} - \frac{\partial \mathbf{j}_k}{\partial q_j}\right)\dot{q}_k\right)^T I_c \boldsymbol{\omega} \\
&\equiv \left(\sum_k \mathbf{z}_{jk}\dot{q}_k\right)^T I_c \boldsymbol{\omega} \qquad (25)
\end{aligned}$$

Now let us evaluate the term denoted by $\mathbf{z}_{jk}$. Consider the skew symmetric form:

$$\begin{aligned}
[\mathbf{z}_{jk}] &= \left[\frac{\partial \mathbf{j}_j}{\partial q_k} - \frac{\partial \mathbf{j}_k}{\partial q_j}\right] = \frac{\partial [\mathbf{j}_j]}{\partial q_k} - \frac{\partial [\mathbf{j}_k]}{\partial q_j} \\
&= \left(\frac{\partial^2 R}{\partial q_j \partial q_k} + \frac{\partial R}{\partial q_j}\frac{\partial R^T}{\partial q_k}\right) - \left(\frac{\partial^2 R}{\partial q_k \partial q_j} + \frac{\partial R}{\partial q_k}\frac{\partial R^T}{\partial q_j}\right) \quad \text{(From Equation 13)} \\
&= \frac{\partial R}{\partial q_j} R^T \left(\frac{\partial R}{\partial q_k} R^T\right)^T - \frac{\partial R}{\partial q_k} R^T \left(\frac{\partial R}{\partial q_j} R^T\right)^T \\
&= -[\mathbf{j}_j][\mathbf{j}_k] + [\mathbf{j}_k][\mathbf{j}_j] \quad \text{(Using the identity } [\mathbf{a}]^T = -[\mathbf{a}]\text{)} \\
&= [\mathbf{j}_k \times \mathbf{j}_j] \quad \text{(Using the identity } [\mathbf{u} \times \mathbf{v}] = [\mathbf{u}][\mathbf{v}] - [\mathbf{v}][\mathbf{u}]\text{)} \\
\Rightarrow \mathbf{z}_{jk} &= \mathbf{j}_k \times \mathbf{j}_j = [\mathbf{j}_k]\mathbf{j}_j \qquad (26)
\end{aligned}$$

Substituting the above in Equation 25, we get:

$$\begin{aligned}
\left(\sum_k \mathbf{z}_{jk}\dot{q}_k\right)^T I_c\boldsymbol{\omega} &= \left(\sum_k [\mathbf{j}_k]\mathbf{j}_j\dot{q}_k\right)^T I_c\boldsymbol{\omega} \\
&= \left(\left(\sum_k [\mathbf{j}_k]\dot{q}_k\right)\mathbf{j}_j\right)^T I_c\boldsymbol{\omega} \\
&= \left(\left[\sum_k \mathbf{j}_k\dot{q}_k\right]\mathbf{j}_j\right)^T I_c\boldsymbol{\omega} \\
&= \left([J_\omega\dot{\mathbf{q}}]\mathbf{j}_j\right)^T I_c\boldsymbol{\omega} = ([\boldsymbol{\omega}]\mathbf{j}_j)^T I_c\boldsymbol{\omega} \\
\text{term 5 (angular)} &= -\mathbf{j}_j^T[\boldsymbol{\omega}]I_c\boldsymbol{\omega} \qquad (27)
\end{aligned}$$

Finally, we substitute the terms computed in Equation 21, Equation 22, Equation 24 and Equation 27 into Equation 20 and rewrite it as:

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} &= (J)_j^T M_c J\ddot{\mathbf{q}} + (J)_j^T M_c \dot{J}\dot{\mathbf{q}} + \mathbf{j}_j^T[\boldsymbol{\omega}]I_c\boldsymbol{\omega} \\
&= \left((J)_j^T M_c J\right)\ddot{\mathbf{q}} + \left((J)_j^T M_c\dot{J} + (J)_j^T[\tilde{\boldsymbol{\omega}}]M_c J\right)\dot{\mathbf{q}} \\
\text{where } [\tilde{\boldsymbol{\omega}}] &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & [J_\omega\dot{\mathbf{q}}] \end{pmatrix} \qquad (28)
\end{aligned}$$

Writing the equations for all the q_j in the vector form, we get:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} = \left(J^T M_c J\right)\ddot{\mathbf{q}} + \left(J^T M_c\dot{J} + J^T[\tilde{\boldsymbol{\omega}}]M_c J\right)\dot{\mathbf{q}} \qquad (29)$$

Derivation using Newton-Euler equations. We can alternatively derive the result in Equation 29 from the Newton-Euler equations in Equation 2. Using Equation 15, we substitute the Cartesian velocities $\mathbf{v}, \boldsymbol{\omega}$ in terms of the generalized velocities $\dot{\mathbf{q}}$ into Equation 2 and get:

$$\begin{aligned} M_c(\dot{J}\dot{\mathbf{q}}) + \begin{pmatrix} \mathbf{0} \\ (J_\omega \dot{\mathbf{q}}) \times I_c J_\omega \dot{\mathbf{q}} \end{pmatrix} &= \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix} \\ \Rightarrow M_c J\ddot{\mathbf{q}} + M_c \dot{J}\dot{\mathbf{q}} + [\tilde{\boldsymbol{\omega}}] M_c J\dot{\mathbf{q}} &= \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix} \end{aligned} \tag{30}$$

From the principle of virtual work in Equation 5, we convert the Cartesian-space forces to the Generalized space by pre-multiplying the above equation with the transpose of the Jacobian J:

$$\left(J^T M_c J\right)\ddot{\mathbf{q}} + \left(J^T M_c \dot{J} + J^T[\tilde{\boldsymbol{\omega}}] M_c J\right)\dot{\mathbf{q}} = J_v^T \mathbf{f} + J_\omega^T \boldsymbol{\tau} \tag{31}$$

The LHS of Equation 31 is identical to the RHS of Equation 29 and they are of the form $M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q}$, where the Mass matrix, the Coriolis and the centrifugal term and the generalized forces are defined as:

$$\begin{aligned} M(\mathbf{q}) &= J^T M_c J \\ C(\mathbf{q}, \dot{\mathbf{q}}) &= (J^T M_c \dot{J} + J^T[\tilde{\boldsymbol{\omega}}] M_c J)\dot{\mathbf{q}} \\ \mathbf{Q} &= J_v^T \mathbf{f} + J_\omega^T \boldsymbol{\tau} \end{aligned} \tag{32}$$

2.3 Articulated rigid body dynamics

We now derive the equations of motion for an open-chain articulated rigid body structure. We follow the derivation of rigid body dynamics in generalized coordinates from Section 2.2.

An articulated rigid body system is represented as a set of rigid bodies connected through joints in a tree structure. Every rigid link has exactly one *parent* joint. The

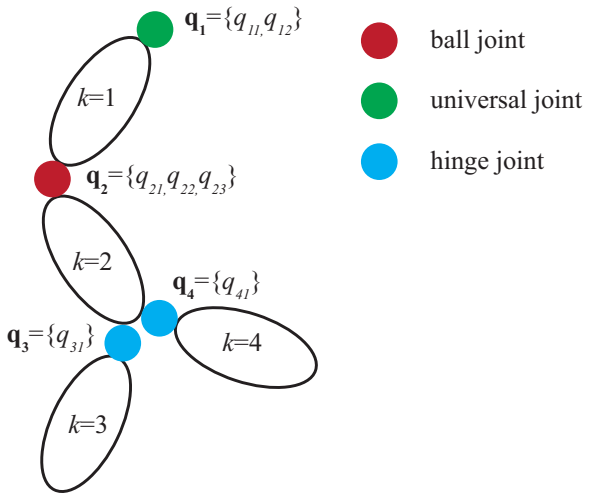


Figure 3: An open-chain articulated rigid body system.

joint corresponding to the root of the tree is special since it does not link the root to any other rigid link. The generalized coordinates are the DOFs of the root link of the tree (that may represent the global translation and rotation), and the joint angles corresponding to the admissible joint rotations for all the other joints.

2.3.1 Definitions

The state of an articulated rigid body system can be expressed as $(\mathbf{x}_k, R_k, \mathbf{v}_k, \boldsymbol{\omega}_k)$, where $k = 1, \cdots, m$ and m is the number of rigid links. Here $\mathbf{x}_k$ and R_k are the position of the COM and the orientation of the rigid link k, and $(\mathbf{v}_k, \boldsymbol{\omega}_k)$ are the linear and angular velocity of the rigid link k viewed in the world frame. Similarly, we define the Cartesian force and torque applied on rigid link k as $(\mathbf{f}_k, \boldsymbol{\tau}_k)$, both of which are expressed in the world frame.

The same articulated rigid body system can be represented in generalized coordinates. We define the generalized state as $(\mathbf{q}, \dot{\mathbf{q}})$, where $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_k, \ldots, \mathbf{q}_m)$ and each $\mathbf{q}_k$ is the set of DOFs of the joint that connects the link k to its parent link (see Figure 3).

We list a few notations and definitions for an articulated rigid body system with m rigid links.

- $p(k)$ returns the index of the parent link of link k. For example in Figure 3, $p(4) = 2$. $p(1, k)$ returns the indices of all the links in the chain from the root to the link k (including k). For example, $p(1, 4) = \{1, 2, 4\}$

- $n(k)$ returns the number of DOFs in the joint that connects the k to the parent link $p(k)$. For example in Figure 3, $n(2) = 3$, $n(3) = 1$ etc. We denote the total number of DOFs in the system by n. e.g. $n = 7$ in Figure 3.

- R_k is the local rotation matrix for the joint k and depends only on the DOFs $\mathbf{q}_k$. R_k^0 is the chain of rotational transformations from the world frame to the local frame of the link k. Therefore, $R_k^0 = R_{p(k)}^0 R_k$. Since the link 1 does not have a parent link, $R_{p(1)}^0 = \mathbf{I}_3$.

2.3.2 Cartesian and generalized velocities

For a single rigid body, Equation 12 and Equation 14 describe the relation between the Cartesian velocities and the generalized velocities. For an articulated rigid body system, we use the same recipe as rigid body dynamics in Section 2.2 and define the Jacobian for each rigid link that relate its respective Cartesian velocities to the generalized velocity of the entire system.

We start with deriving the relation for the angular velocity. The angular velocity of link k viewed in the world frame is:

$$\begin{aligned}
[\boldsymbol{\omega}_k] &= \dot{R}_k^0 {R_k^0}^T = (\dot{R_{p(k)}^0} R_k)(R_{p(k)}^0 R_k)^T \\
&= (\dot{R}_{p(k)}^0 R_k + R_{p(k)}^0 \dot{R}_k) R_k^T {R_{p(k)}^0}^T \\
&= \dot{R}_{p(k)}^0 {R_{p(k)}^0}^T + R_{p(k)}^0 \left(\dot{R}_k R_k^T\right) {R_{p(k)}^0}^T \equiv [\boldsymbol{\omega}_{p(k)}] + R_{p(k)}^0 [\hat{\boldsymbol{\omega}}_k] {R_{p(k)}^0}^T \qquad (33)
\end{aligned}$$

In the above equation, we define $[\hat{\boldsymbol{\omega}}_k] = \dot{R}_k R_k^T$ that denotes the angular velocity of the link k in the frame of its parent link $p(k)$ since the rotation matrix R_k is the rotation of the rigid link k with respect to $p(k)$. We can further write $\hat{\boldsymbol{\omega}}_k = \hat{J}_{\omega k} \dot{\mathbf{q}}_k$ where $\hat{J}_{\omega k}$

is the *local* Jacobian matrix that relates the joint velocity of link k to its angular velocity in the frame of the parent link $p(k)$. The dimension of $\hat{J}_{\omega k}$ is $3 \times n(k)$.

Using the property of skew symmetric matrix, $[R\boldsymbol{\omega}] = R[\boldsymbol{\omega}]R^T$, we can express Equation 33 in the vector form as:

$$\begin{aligned} \boldsymbol{\omega}_k &= \boldsymbol{\omega}_{p(k)} + R^0_{p(k)} \hat{J}_{\omega k} \dot{\mathbf{q}}_k \\ &= \sum_{l \in p(1,k)} R^0_{p(l)} \hat{J}_{\omega l} \dot{\mathbf{q}}_l \quad \text{(By unrolling the recursive definition)} \\ &\equiv J_{\omega k} \dot{\mathbf{q}} \end{aligned} \tag{34}$$

where the Jacobian $J_{\omega k}$ is:

$$J_{\omega k} = \begin{pmatrix} \hat{J}_{\omega 1} & \dots & R^0_{p(l)} \hat{J}_{\omega l} & \dots & \mathbf{0} & \dots \end{pmatrix} \tag{35}$$

Note that the zero matrices $\mathbf{0}$ of size $3 \times n(l)$ in $J_{\omega k}$ correspond to joint DOFs $\mathbf{q}_l$ that are *not* in the chain of transformations from the root to the link k. Let us look at a couple of examples using the articulated rigid body system in Figure 3:

$$\begin{aligned} \boldsymbol{\omega}_1 &= (\hat{J}_{\omega 1} \;\; \mathbf{0} \;\; \mathbf{0} \;\; \mathbf{0}) \dot{\mathbf{q}} \\ \boldsymbol{\omega}_4 &= (\hat{J}_{\omega 1} \;\; R^0_1 \hat{J}_{\omega 2} \;\; \mathbf{0} \;\; R^0_2 \hat{J}_{\omega 4}) \dot{\mathbf{q}} \end{aligned}$$

where $\hat{J}_{\omega 1} \in \Re^{3\times 2}$, $\hat{J}_{\omega 2} \in \Re^{3\times 3}$ and $\hat{J}_{\omega 4} \in \Re^{3\times 1}$. Depending on the representation of the rotation $\mathbf{q}_k$, $\hat{J}_{\omega k}$ can assume different values. For example, if the joint between link 1 and link 2 in Figure 3 is represented as three Euler rotations, $R^{(x)}$, $R^{(y)}$, and $R^{(z)}$ such that $R_2(\mathbf{q}_2) = R^{(x)}(q_{21})R^{(y)}(q_{22})R^{(z)}(q_{23})$, we have:

$$\hat{J}_{\omega 2} = \begin{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} & R^{(x)} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} & R^{(x)} R^{(y)} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix} \tag{36}$$

If the joint is represented as a quaternion or an exponential map, $\hat{J}_{\omega k}$ does not have a simple form. As an example, the relation between the rotation matrix R_k and the

exponential map representation $\mathbf{q}_k = (q_{k1}, q_{k2}, q_{k3})$ can be written as:

$$R_k(\mathbf{q}_k) = e^{[\mathbf{q}_k]} = \mathbf{I}_3 + \frac{sin\theta}{\theta}[\mathbf{q}_k] + \frac{1 - cos\theta}{\theta^2}[\mathbf{q}_k]^2 \tag{37}$$

where $\theta = ||\mathbf{q}_k||$. The Jacobian $\hat{J}_{\omega k}$ can be derived by equating the result of $\dot{R}_k R_k^T$ to $[\hat{J}_{\omega k}\dot{\mathbf{q}}_k]$:

$$\begin{aligned} \hat{J}_{\omega k} &= R_k \left(\mathbf{I}_3 - \frac{1 - cos\theta}{\theta^2}[\mathbf{q}_k] + \frac{\theta - sin\theta}{\theta^3}[\mathbf{q}_k]^2 \right) \\ &= \mathbf{I}_3 + \frac{1 - cos\theta}{\theta^2}[\mathbf{q}_k] + \frac{\theta - sin\theta}{\theta^3}[\mathbf{q}_k]^2 \end{aligned} \tag{38}$$

For the case when $\theta \to 0$, R_k and $\hat{J}_{\omega k}$ can be approximated as follows:

$$R_k = \mathbf{I}_3 + [\mathbf{q}_k] + \frac{1}{2}[\mathbf{q}_k]^2 \tag{39}$$

$$\hat{J}_{\omega k} = \mathbf{I}_3 + \frac{1}{2}[\mathbf{q}_k] + \frac{1}{6}[\mathbf{q}_k]^2 \tag{40}$$

Similar to the angular velocity, the linear velocity of the center of mass of the link k can be expressed in terms of the generalized velocity:

$$\mathbf{v}_k = J_{vk}\dot{\mathbf{q}}, \quad \text{where } J_{vk} = \frac{\partial \mathbf{x}_k}{\partial \mathbf{q}} = \frac{\partial W_k^0 \mathbf{c}_k}{\partial \mathbf{q}}. \tag{41}$$

where the chain of homogeneous transformations from the world frame to the local frame of link k is denoted as W_k^0. Note that W_k^0 is different from R_k^0 in that W_k^0 includes the translational transformations. $\mathbf{c}_k$ is a constant vector that denotes the center of mass of link k in its local frame.

We can concatenate the Cartesian velocities into a single vector $\mathbf{V}_k$ and denote the relation as:

$$\begin{aligned} \mathbf{V}_k &= J_k\dot{\mathbf{q}} \\ \text{where} \qquad \mathbf{V}_k &= \begin{pmatrix} \mathbf{v}_k \\ \boldsymbol{\omega}_k \end{pmatrix} \text{ and } J_k = \begin{pmatrix} J_{vk} \\ J_{\omega k} \end{pmatrix} \end{aligned} \tag{42}$$

2.3.3 Equations of motion in generalized coordinates

We now derive the equations of motion of an articulated rigid body system in generalized coordinates. The kinetic energy T of the entire system can be expressed as the sum of kinetic energies of all the rigid links as $T = \sum_k T_k$. Therefore the equations of motion of the system can be computed as:

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} &= \frac{d}{dt}\left(\frac{\partial \sum_k T_k}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial \sum_k T_k}{\partial \mathbf{q}} \\
&= \sum_k \left(\frac{d}{dt}\left(\frac{\partial T_k}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T_k}{\partial \mathbf{q}}\right) \\
&= \sum_k \left(\left(J_k^T M_{ck} J_k\right)\ddot{\mathbf{q}} + \left(J_k^T M_{ck}\dot{J}_k + J_k^T[\tilde{\boldsymbol{\omega}}_k] M_{ck} J_k\right)\dot{\mathbf{q}}\right) \\
&= \sum_k \left(J_k^T M_{ck} J_k\right)\ddot{\mathbf{q}} + \sum_k \left(J_k^T M_{ck}\dot{J}_k + J_k^T[\tilde{\boldsymbol{\omega}}_k] M_{ck} J_k\right)\dot{\mathbf{q}} \\
&\equiv M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) \qquad (43)
\end{aligned}$$

In deriving the above equation, we use the equations of motion in generalized coordinates for a single rigid body defined in Equation 29 subscripted by k for the dynamics of k^{th} link in the multi body system. The Jacobian J_k for the k^{th} link is defined in Equation 42.

2.4 Conversion between Cartesian and generalized coordinates

In practice, we often want to use third-party rigid body simulators rather than develop our own. There are a few widely used physics engines that provide efficient, robust, and fairly accurate rigid body simulation and collision handling. Open Dynamics Engine [5], PhysX [6], Havok [3] and Bullet Physics [2] are perhaps the most popular choices among game developers and academic researchers. These commercial simulators use the maximal representation rather than generalized coordinates described above. That is, these simulators represent each link in the articulated rigid body system as six DOFs, leading to a redundant system with additional constraints

between links. A common practice is to develop control algorithms in generalized coordinates and do forward simulation using a commercial physics engine, such as ODE. This requires some conversion between Cartesian and generalized coordinates.

2.4.1 Velocity conversion

We can concatenate all $2m$ Jacobian matrices corresponding to each link into a single Jacobian that relates the generalized velocity to the Cartesian velocity of each link:

$$\mathbf{V} \equiv \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_m \\ \boldsymbol{\omega}_1 \\ \vdots \\ \boldsymbol{\omega}_m \end{pmatrix} = \begin{pmatrix} J_{v1} \\ \vdots \\ J_{vm} \\ J_{\omega 1} \\ \vdots \\ J_{\omega m} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{q}}_1 \\ \vdots \\ \dot{\mathbf{q}}_m \end{pmatrix} \equiv \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{\mathbf{q}} \equiv J\dot{\mathbf{q}} \tag{44}$$

Typically, the Jacobian J is full column rank because the number of DOFs in the maximal representation is more than that in the generalized representation, i.e $6m > n$. To compute $\dot{\mathbf{q}}$ from $\mathbf{V}$, we will end up solving a over-constrained linear system. We can use pseudo inverse of J to compute $\dot{\mathbf{q}}$:

$$\dot{\mathbf{q}} = J^{+}\mathbf{V} \tag{45}$$

where the pseudo-inverse notation $J^{+} = (J^T J)^{-1} J^T$. If this least-square solution does not exactly solve the linear system (i.e. $J\dot{\mathbf{q}} = \mathbf{V}$), it indicates that $\mathbf{V}$ cannot be achieved in the generalized coordinates without violating constraints of the system (e.g. constraints that keep links connected).

Computing J^{+} may be expensive for a system with many rigid links. Alternatively, we can rewrite the equation using the relative velocity between a child and a parent link expressed in the local frame of the parent, instead of using velocities of each link expressed in the world frame. As an example, we write the simplified expression for

the angular velocity of link k using Equation 34 as:

$$\begin{aligned} \boldsymbol{\omega}_k - \boldsymbol{\omega}_{p(k)} &= R^0_{p(k)}\hat{\boldsymbol{\omega}}_k = R^0_{p(k)}\hat{J}_{\omega k}\dot{\mathbf{q}}_k \\ \Rightarrow (-\mathbf{I}_3 \;\; \mathbf{I}_3) \begin{pmatrix} \boldsymbol{\omega}_{p(k)} \\ \boldsymbol{\omega}_k \end{pmatrix} &= R^0_{p(k)}\hat{J}_{\omega k}\dot{\mathbf{q}}_k \end{aligned} \tag{46}$$

Combining these equations for all the links, we get:

$$\begin{aligned} D\boldsymbol{\omega} = DJ_\omega\dot{\mathbf{q}} &= \text{blockdiag}(\hat{J}_{\omega 1}, \ldots, R^0_{p(m)}\hat{J}_{\omega m})\dot{\mathbf{q}} \\ &= \text{blockdiag}(\mathbf{I}_3, \ldots, R^0_{p(m)})\text{blockdiag}(\hat{J}_{\omega 1}, \ldots, \hat{J}_{\omega m})\dot{\mathbf{q}} \\ &\equiv R\hat{J}_\omega\dot{\mathbf{q}} \end{aligned} \tag{47}$$

where D is a constant matrix that encodes the connectivity between links. For example, matrix D for the system in Figure 3 looks like:

$$D = \begin{pmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I}_3 & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_3 & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_3 & \mathbf{0} & \mathbf{I}_3 \end{pmatrix} \tag{48}$$

The relations between $\hat{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}$, and $\hat{J}_\omega$ and J_ω follow from Equation 47:

$$\begin{aligned} \hat{\boldsymbol{\omega}} &= R^T D\boldsymbol{\omega} \\ \text{and } \hat{J}_\omega &= R^T DJ_\omega \end{aligned} \tag{49}$$

The matrix $\hat{J}_\omega$ being block diagonal is much sparser as compared to J_ω.

If $\mathbf{q}$ satisfies the over-constrained system of equations $\mathbf{V} = J\dot{\mathbf{q}}$, using any n independent constraints out of $6m$ to solve this linear system will result in the same $\dot{\mathbf{q}}$. This can be explained by the problem of fitting an unknown plane to $6m$ 3D points as $A\mathbf{x} = \mathbf{b}$, where $A \in \Re^{6m\times 3}$. If all $6m$ points happen to lie on a plane, i.e. there exists an $\mathbf{x}$ that exactly satisfies the over-constrained system, any three distinctive points we pick as the constraints will result in the same plane. Therefore, if we know $\mathbf{V}$ can

be achieved in the generalized coordinates, we can pick a subset of rows from J to form a J' such that the rank of J' is n, and compute $\dot{\mathbf{q}} = J'^{+}\mathbf{V}'$, where $\mathbf{V}'$ are the velocity components corresponding to the rows in J'. The solution $\dot{\mathbf{q}}$ to this system will be the same for any J'.

For a system with only rotational DOFs, it is sufficient to invert only J_ω which is also a full column rank matrix ($J_\omega \in \Re^{3m \times n}$ and $n \leq 3m$). This is because each rotational joint can have at most three independent DOFs. We then can compute the velocities of the rotational DOFs $\dot{\mathbf{q}}$ as:

$$\begin{aligned} \dot{\mathbf{q}} &= J_\omega^{+}\boldsymbol{\omega} \\ &= \hat{J}_\omega^{+} R^T D\boldsymbol{\omega} = \hat{J}_\omega \hat{\boldsymbol{\omega}} \\ &= \text{blockdiag}\left(\hat{J}_{\omega 1}^{+}, \ldots, \hat{J}_{\omega m}^{+}\right) \hat{\boldsymbol{\omega}} \\ \text{or } \dot{\mathbf{q}}_k &= \hat{J}_{\omega k}^{+} \hat{\boldsymbol{\omega}}_k \quad , k \in 1 \ldots m \end{aligned} \tag{50}$$

From this formulation, we see that the problem of computing pseudo-inverse of a matrix J_ω is reduced to computing m pseudo-inverses of much smaller constant-sized matrices $\hat{J}_{\omega k}$. Note that $\dot{\mathbf{q}}$ computed in Equation 50 also satisfies the linear velocity relation $\mathbf{v} = J_v \dot{\mathbf{q}}$.

For systems that include translation DOFs as well, we can separately solve for the rotational DOFs as in Equation 50 and solve for the translational DOFs for any link k as $\dot{\mathbf{q}}_k = J_{vk}^{+}\mathbf{v}_k$. In most of the cases, only the root joint has translational DOFs making the computation of generalized translational velocities extremely simple as J_{v1} becomes an identity matrix.

2.4.2 Force conversion

The relation between the Cartesian force and the generalized force can be found in Equation 5:

$$\mathbf{Q} = \sum_k {J_{vk}^{'}}^T \mathbf{f}_k + J_{\omega k}^T \boldsymbol{\tau}_k^{'} = \begin{pmatrix} {J_v^{'}}^T & J_\omega^T \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau}^{'} \end{pmatrix}, \quad \text{where } J_{vk}^{'} = \frac{\partial \mathbf{r}_k}{\partial \mathbf{q}} \tag{51}$$

where $\mathbf{r}_k$ is the point of application of the Cartesian force $\mathbf{f}_k$ and $\boldsymbol{\tau}_k^{'}$ is the *body torque* applied to link k expressed in the world frame.

Note. Body torque $\boldsymbol{\tau}_k^{'}$ is the torque applied on link k in the world frame and does *not* include the torque induced by the linear forces $\mathbf{f}_i$. However, the definition for torque in Equation 32 *includes* the torque $[\mathbf{r}_k - \mathbf{x}_k]\mathbf{f}_i$ due to each force $\mathbf{f}_k$ ($\mathbf{x}_k$ is the COM of the link i). As a result, the linear Jacobian J_v in Equation 32 is defined for the COM of the respective rigid link and $J_v^{'}$ in Equation 51 is defined for the point of application of the force. It is easy to verify that ${J_{vk}^{'}}^T \mathbf{f}_k = J_{vk}^T \mathbf{f}_k + J_{\omega k}^T [\mathbf{r}_k - \mathbf{x}_k]\mathbf{f}_k$. i.e. $\boldsymbol{\tau}_k = \boldsymbol{\tau}_k^{'} + [\mathbf{r}_k - \mathbf{x}_k]\mathbf{f}_k$.

Often many controllers (such as a tracking controller) find it convenient to compute the Cartesian-space *joint torques* in the local frame of the parent link rather than body torques in the world frame. Joint torque $\hat{\boldsymbol{\tau}}_k$ in the frame of parent link $p(k)$ is defined such that positive torque in the world frame $R_{p(k)}^0 \hat{\boldsymbol{\tau}}_k$ is applied to the link k and negative torque $-R_{p(k)}^0 \hat{\boldsymbol{\tau}}_k$ is applied to the parent link $p(k)$. Therefore, the body torque $\boldsymbol{\tau}_k^{'}$ applied to the link k can be written in terms of the joint torques as $\boldsymbol{\tau}_k^{'} = R_{p(k)}^0 \hat{\boldsymbol{\tau}}_k - \sum_l R_k^0 \hat{\boldsymbol{\tau}}_l$, $\forall l : k = p(l)$. Collecting the body torques for all the rigid links in the vector $\boldsymbol{\tau}$, the relation between the body torques and the joint torques can be defined as:

$$\boldsymbol{\tau}^{'} \quad = \quad D^T R \hat{\boldsymbol{\tau}} = (R^T D)^T \hat{\boldsymbol{\tau}} \tag{52}$$

where R, D are defined in Equation 49. We now substitute Equation 52 in Equation 51

and get:

$$\begin{aligned}
\mathbf{Q} &= \begin{pmatrix} {J_v'}^T & J_\omega^T \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ (R^T D)^T \hat{\boldsymbol{\tau}} \end{pmatrix} \\
&= \begin{pmatrix} {J_v'}^T & (R^T D J_\omega)^T \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \hat{\boldsymbol{\tau}} \end{pmatrix} \\
&= \begin{pmatrix} {J_v'}^T & \hat{J}_\omega^T \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \hat{\boldsymbol{\tau}} \end{pmatrix} \quad \text{(Using Equation 49)} \qquad (53)
\end{aligned}$$

Equation 53 gives the relation to convert the given Cartesian forces $\mathbf{f}$ and joint torques $\hat{\boldsymbol{\tau}}$ to the generalized forces $\mathbf{Q}$.

We now describe the process to convert the given generalized forces $\mathbf{Q}$ to Cartesian forces and torques. In general, the transposed Jacobian in Equation 51 can be inverted using pseudo-inverse to get the Cartesian forces and torques. Note that the relation represents an under-constrained system when solving for $\mathbf{f}$ and $\boldsymbol{\tau}'$. This is because the size of the unknowns is $6m$ and the number of constraints are n with $n \leq 6m$. Therefore, we get particular solutions for the Cartesian forces and torques out of many possible solutions.

Based on the information about the form of $\mathbf{Q}$, we can solve for the Cartesian forces and torques in different ways. We describe the solutions to the following cases:

1. **General case.** In the most general case, the points of application of the Cartesian forces are not known. Therefore, we cannot use the Jacobian J_v' in Equation 51. This forces us to assume the points of application to be the COM of each link and compute the torques $\boldsymbol{\tau}$ instead of body torques $\boldsymbol{\tau}'$. i.e. , we can invert the transposed Jacobian by computing its pseudo-inverse and a get a particular least squared solution for $\mathbf{f}$ and $\boldsymbol{\tau}$ as:

$$\begin{pmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix} = \begin{pmatrix} {J_v}^T & J_\omega^T \end{pmatrix}^+ \mathbf{Q} \qquad (54)$$

If the points of the force application are known, the Jacobian in Equation 53 can be inverted to obtain the forces $\mathbf{f}$ and the joint torques $\hat{\boldsymbol{\tau}}$.

2. **No linear forces.** The more common case for many controllers involves the conversion of only the joint torques from generalized to the Cartesian coordinates. Therefore, the linear forces $\mathbf{f}$ are zero and Equation 53 can be simplified further to result in the following conversion relation:

$$\begin{aligned} \hat{\boldsymbol{\tau}} &= (\hat{J}_{\omega}^{T})^{+}\mathbf{Q} \\ \text{or } \hat{\boldsymbol{\tau}}_k &= (\hat{J}_{\omega k}^{T})^{+}\mathbf{Q}_k \;\; \forall k \in 1 \ldots m \end{aligned} \tag{55}$$

where $\mathbf{Q}_k$ denotes the components of the generalized forces corresponding to the rotational DOFs $\mathbf{q}_k$. Note that the size of the matrix $\hat{J}_{\omega k}^{T}$ is $n(k) \times 3$ and $n(k) \leq 3$. This implies that we get a particular least squared solution for each $\hat{\boldsymbol{\tau}}_k$ out of possibly many solutions that would give rise to the same $\mathbf{Q}_k$ using the relation $\mathbf{Q}_k = \hat{J}_{\omega k}^{T}\hat{\boldsymbol{\tau}}_k$.

CHAPTER III

OPTIMIZATION-BASED INTERACTIVE MOTION SYNTHESIS

In this chapter, we present a physics-based approach to synthesizing motion of a virtual character in a dynamically varying environment [41]. Our approach views the motion of a responsive virtual character as a sequence of solutions to the constrained optimization problem formulated at every time step. This framework allows the programmer to specify active control strategies using intuitive kinematic goals, significantly reducing the engineering effort entailed in active body control. Our optimization framework can incorporate changes in the character's surroundings through a synthetic visual sensory system and create significantly different motions in response to varying environmental stimuli. Our results show that our approach is general enough to encompass a wide variety of highly interactive motions.

3.1 Introduction

To date, animating the behavior of human characters in a dynamic environment remains primarily an animator-driven activity. Unlike simulation of passive phenomenon such as smoke, water, and clothing where automated algorithms have seen wide commercial adoption, the reaction of a virtual human character depends largely on the interaction between her own goals and environmental factors, in addition to the laws of physics. For example, when losing balance a real person will reposition her body to slow the fall while grabbing onto any nearby object that appears stable. Even for such a simple task, the sheer scale of possible objects and environments

a character can interact with makes designing a generic simulation algorithm challenging. Consequently to date, character animation is still primarily done through key-framing or blending pre-recorded motion sequences.

Physical simulation via robotic controllers has the potential to be a general framework for simulating believable character interactions without the need of extensive data or user effort. In the past, specialized control algorithms have proven capable of generating diverse motions such as balancing, running, and diving. Despite these successes, robotics controllers exhibit two main drawbacks. First, designing robotics controllers is a difficult and time consuming process. Good controller design requires modeling of the musculoskeletal system and tuning of model parameters that have nonlinear relationships with the output motion. Second once designed, controllers are often brittle, only working under a narrow range of conditions. Changes in the environment often necessitate significant tuning of control parameters or a redesign of the controller itself.

We explore an alternative framework for active character simulation, physics based optimization, which formulates motion synthesis as an optimization problem. Up to now, physics based optimization has been applied to generating motions in pre-planned situations where all the constraints and objectives are known a priori. Within this domain, the framework has proven capable of synthesizing a wide class of realistic human motions from walking to complex gymnastics. In addition, user control of the animation is straightforward. To specify a motion, the programmer only needs to describe the goals of the motion (e.g. jump to this position). The optimization framework handles how the motion is achieved. These traits make physics-based optimization an appealing framework to synthesize character animations.

This chapter describes a physics-based optimization framework for *interactive* character animation. In our system, the motion of responsive virtual character is a sequence of solutions to a constrained optimization problem formulated at every

time step. In this framework, the controller is a process that directs the motion by providing kinematic goals, such as desired body position or velocity, into the optimization problem at each time step. Instead of explicitly solving for internal joint torques and numerically integrating them to solve for motion, our approach directly optimizes the joint configurations subject to the laws of physics, environment constraints, bio-mechanical limitations, and task-level control strategies. When the character is in contact with the environment, we also explicitly optimize the contact forces to achieve desired tasks while maintaining physical realism. For a passive dynamic system, there is no benefit in using the optimization to solve for the motion, since the problem is well constrained and can be solved efficiently by a standard forward simulator. The real advantages of our method are exposed when simulating an active dynamic system in sustained contact with the environment, such as most everyday human activities. The unknown actuation and contact forces in such a dynamic system pose an under-determined problem. By solving the actuation (implicitly), contact forces, and the final motion all in one procedure, our method allows the control policies to be intuitively formulated into functions of joint configurations without referring to forces and torques.

In this chapter, we demonstrate several benefits that arise from casting interactive motion synthesis as an optimization problem. First, the programmer can specify active control strategies through a controller using kinematic goals, thus retaining the intuitive user-level control that optimization offers in offline motion synthesis. Second, the programmer can compose complex control strategies by combining simple control strategies in a finite-state machine-like structure. We demonstrate a versatile virtual character coping with a series of unexpected disturbances using a combination of control strategies in an autonomous fashion (Figure 13). Third, the character can perceive changes in the environment through a synthetic visual sensory system. Our optimization incorporates this information along with other high-level decisions

as additional objectives and constraints. Consequently, the same controller creates significantly different motions in response to different environmental stimuli. As examples, we demonstrate a character that can use environment features to retain her balance while dodging flying objects. Finally, as our results show, this framework is general enough to encompass a wide variety of highly interactive motions. We demonstrate that from a simple balance controller, to wall climbing, and gymnastics.

3.2 Related work

Designing a virtual human character that actively responds to the physical environment is a long standing challenge in computer animation. The variational optimization based approach directly solves for the entire motion trajectory according to energy consideration and user specifications. Solving for nonlinear dynamic constraints and energy-based objective functions produces good results on simple skeletons [95, 18, 60], as well as on abstract human models [72] with simplified dynamics [59, 29]. With the aid of motion data, researchers have formulated the optimization problem in a reduced space biased towards natural human motion [75, 85], or extracted parameters from the data that capture muscle preferences and joint stiffness [58]. The optimization approach allows the programmer to describe the motion task by providing keyframe-like constraints in the joint space. However, standard optimization-based approaches are not suited for interactive applications because all the constraints and objectives need to be specified a priori. Our method treats every simulation time step as an independent optimization problem with a new set of constraints and objectives. Any unscripted events in the current time step, such as user input or collisions, will be responded to appropriately in the next time step.

Active body control with physical simulation presents many obvious advantages over optimal trajectory approach in the domain of creating responsive virtual characters. Researchers have designed basic balance controllers for bipedal systems [73,

50, 89, 79, 7, 49], as well as more versatile motion such as running, vaulting, and cycling [38]. Yin et al. reduce control design to a few kinematics poses with the help of a robust balance strategy for walking and running [105]. Wooten and Hodgins [96] concatenated a sequence of transition controllers that generate successive motion sequences. Faloutsos et al. [28] demonstrated that a virtual character can be simulated by composing multiple primitive robotic controllers. Several companies have successfully applied similar technologies to commercial products by providing a repertoire of motor skills [4]. The specific details regarding their implementation are unknown, but it is likely that each individual controller requires fine tuning of the physical parameters. To synthesize animations involving interactions with the environment, they rely on users to establish contact constraints at the right timing. Robotic controller simulation yields physically plausible motion, often in real-time, but requires an expert to tune the parameters properly. Our work provides a generic framework for rapid designing active control procedures that require minimal physical parameter tuning, yielding an adaptable controller for characters of arbitrary design.

To circumvent the issues of over-specialization, many researchers suggested exploiting online, local optimization techniques that adjust the current dynamic parameters to new situations [7, 100, 83, 82]. Our method is inspired by the same idea, but instead of adjusting the parameters in the force domain and obtaining the motion by numerical integration methods, we directly optimize the joint configurations according to the control policies. By directly controlling the joint configuration instead of joint acceleration, we can formulate constraints that are satisfied exactly in the joint space without numerical errors due to the integration. Furthermore, we do not require the constraints and their first derivative to be satisfied initially. This flexibility allows us to arbitrarily add or change constraints in the position space.

Much previous work in robotics has addressed the problem of controlling multiple tasks for robots or manipulators [55, 63, 77, 76]. In computer graphics, Abe and

Popović [8] demonstrated a prioritized control approach that allows a virtual character to execute tasks at different priority levels without interfering with a posture tracking controller. They later proposed an optimization-based control that formulates the multiple objectives into a quadratic programming problem. This formulation allows for a compromise between several conflicting objectives, such as balancing and pose tracking [7]. Our framework also addresses the problem of multiple objectives by formulating an optimization. Instead of solving for the control, however, we directly solve for motion that achieves the coordination among multiple objectives. The weights of the objectives directly influence the task priority in the motion without interference from other physical parameters. Furthermore, because we use a constrained optimization to solve for motion, we can formulate a primary task that can never be violated as a constraint instead of an objective.

Using keyframe-like control to create physically responsive animation provides a practical tool for many computer animation applications [39, 83]. Our simple framework for specifying control strategies is inspired by earlier systems designed by Stewart and his colleagues [82]. Their proposed control schemes allows the programmer to control any linear combination of the state variables in the second derivative domain, such as the acceleration of the center of mass. Our optimization formulation further allows the programmer to control variables themselves, rather than a more complicated second derivative domain. Liu [56] described a similar optimization framework for synthesis of hand animation based on specifications of the manipulated objects. She demonstrated that simple grasping-like tasks can be produced with a few keyframes on the object. Our approach addresses more complicated issues such as postural balance and coordination in full body motion. In addition, our approach aims for designing an autonomous dynamic system by incorporating the sensory information to the control system.

3.3 Overview

We view responsive character motion as a sequence of solutions to a constrained optimization formulated at every time step. Each optimization yields an optimal joint configuration based on the user-specified goals and energy consideration, subject to the laws of physics. We breakdown our motion synthesis framework into following main components:

1. *Motion Synthesizer* : The motion synthesizer forms the core of the framework. Given the current dynamic state of the character, motion synthesizer formulates an optimization that solves for the joint configuration of the next time step. To ensure physical realism in the synthesized motion , we enforce Lagrange's equations of motion and Coulomb's friction model as constraints and minimize the change of muscle force usage as objective in the optimization.

2. *User-specified Controller* : To synthesize an active character behavior, the controller adds kinematic goals to the objective function of the optimization at current time step. The programmer creates this controller by specifying the goals conforming to the character's internal dynamic state and/or the external environmental state.

3. *Environment Knowledge from Visual Sensory* : Complex control strategies often depend on sensory inputs the character gathers from the environment. We endow our virtual characters with a synthetic visual sensor and allow the programmer to formulate control strategies that depend on the sensor input.

4. *User Interaction* : The user can interact with the on-going character motion by applying external forces or adding kinematic constraints or objective in an interactive fashion.

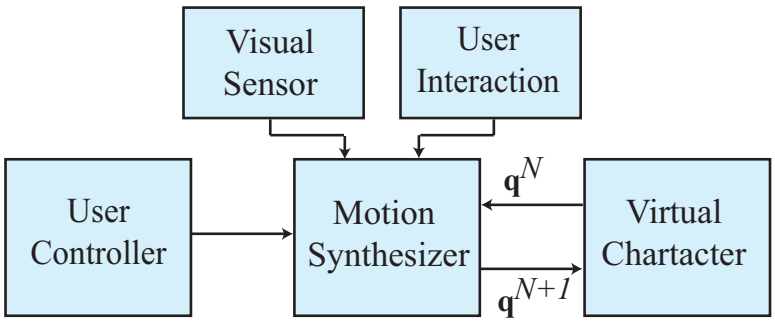


Figure 4: The controller provides control strategies to the motion synthesizer which takes in the current dynamic state of the character, $\mathbf{q}^N$, and outputs new joint configuration, $\mathbf{q}^{N+1}$, for the next time step. Additional inputs can be provided to the synthesizer in the form of user interactions and environment knowledge through visual sensory system.

Figure 4 illustrates the relationship between the components described above. At each time step, the user-specified controller formulates appropriate objectives and constraints based on the current dynamic state of the character and the environment information from the visual sensory system. The motion synthesizer formulates an optimization problem comprising the controller-generated objective and constraints, external disturbances, and objectives and constraints enforcing physical realism. The solution to the optimization problem yields the character's joint configuration for the next time step.

3.4 Optimization setup for motion synthesis

The heart of our framework is the formulation of an optimization problem to synthesize the character's motion at each time step. Given physical constraints and objectives specified by the programmer, we solve for character's joint configuration for the next time step and external contact forces simultaneously in the optimization. Solving for external contact forces is equivalent to determining how much force the character applies at the point of contact. We do not solve for internal joint muscles explicitly in the optimization.

We represent the character's skeleton as a transformation hierarchy of 18 body

nodes with 31 degrees of freedom (DOFs) in reduced coordinates representing joints and 6 DOFs representing the global translation and rotation.

To enforce physical realism in the synthesized motion, we formulate Lagrange's equations of motion as constraints in our optimization. Lagrange's equations are reformulation of Newton's equations of motion in generalized coordinates (DOFs in our case). We enforce Lagrange's constraint L_j on each DOF q_j of the root of the skeleton's hierarchy (see Equation 6):

$$L_j(\mathbf{q}, \boldsymbol{\lambda}) = \sum_{i \in N(j)} \left(\frac{d}{dt} \frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} \right) - Q_j^g - Q_j^c(\boldsymbol{\lambda}) - Q_j^{ext} = 0 \tag{56}$$

where $\mathbf{q} \equiv (q_0, q_1, \ldots)^T$ is a vector of all DOFs and $\boldsymbol{\lambda}$ are the parameters of contact forces. Q_j^g, Q_j^c, and Q_j^{ext} represent the gravitational force, contact force, and an additional external force respectively in generalized coordinates.

The generalized forces Q_j^g and Q_j^{ext} correspond to the gravity force $\sum m_i \mathbf{g}$ and the user input force $\mathbf{F}_{ext}$. The force $\mathbf{F}_{ext}$ acts on the body node k at point $\mathbf{p}_k$ in its local coordinate frame. The generalized forces are given by:

$$Q_j^g = \sum_{i \in N(j)} \left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \right) . (m_i \mathbf{g}) \tag{57}$$

$$Q_j^{ext} = \left(\frac{\partial \mathbf{W}_k}{\partial q_j} \mathbf{p}_k \right) . \mathbf{F}_{ext} \tag{58}$$

where $\mathbf{c}_i$ is the center of mass (COM) of body node i in its local coordinate frame.

The first two terms in Equation 56 measure the inertia force due to the acceleration of DOF q_j in generalized coordinates. T_i denotes the kinetic energy of body node i and $N(j)$ is the set of body nodes in the subtree of DOF q_j. In the transformation hierarchy, the inertia force of node i due to the DOF q_j can be computed as:

$$\frac{d}{dt} \frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} = \mathrm{tr} \left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^{\ T} \right) \tag{59}$$

where $\mathrm{tr}\,()$ gives the trace of a matrix and $\mathbf{W}_i$ is the chain of homogeneous transformations from the root node to body node i. $\mathbf{M}_i$ denotes the mass tensor of the body

node i defined as:

$$\mathbf{M}_i \equiv \int\int\int \rho \mathbf{x}\mathbf{x}^T dx\ dy\ dz \tag{60}$$

where an infinitesimal point $\mathbf{x} \equiv (x, y, z, 1)^T$ in the local coordinates of body node i has mass density ρ.

Because we represent time as discrete samples, all the functions of time-varying variables need to be represented in a discrete domain. We discretize the time into samples with small intervals Δt. We define the velocity and the acceleration of a DOF q_j, at current time sample N, by central finite differences:

$$\dot{q_j}^N \equiv \frac{q_j^{N+1} - q_j^{N-1}}{2\Delta t} \tag{61}$$

$$\ddot{q_j}^N \equiv \frac{q_j^{N+1} - 2q_j^N + q_j^{N-1}}{\Delta t^2} \tag{62}$$

For a body node i, $\dot{\mathbf{W}}_i$ and $\ddot{\mathbf{W}}_i$ at time sample N can be written as:

$$\dot{\mathbf{W}}_i = \frac{\partial \mathbf{W}_i}{\partial \mathbf{q}}\dot{\mathbf{q}} = \sum_j \frac{\partial \mathbf{W}_i}{\partial q_j}\dot{q}_j \tag{63}$$

$$\ddot{\mathbf{W}}_i = \sum_j \left(\sum_k \Big(\frac{\partial^2 \mathbf{W}_i}{\partial q_k \partial q_j}\dot{q}_k\Big)\dot{q}_j + \frac{\partial \mathbf{W}_i}{\partial q_j}\ddot{q}_j \right) \tag{64}$$

For clarity, we drop the superscript henceforth, for quantities at time sample N. The derivative terms $\frac{\partial \mathbf{W}_i}{\partial q_j}$ and $\frac{\partial^2 \mathbf{W}_i}{\partial q_k \partial q_j}$ can be computed analytically, since $\mathbf{W}_i$ is a differentiable function of $\mathbf{q}$.

In this discrete formulation, the optimization at each time step N solves for the DOFs at the next time step, $\mathbf{q}^{N+1}$, given the current and previous DOFs, $\mathbf{q}^N$ and $\mathbf{q}^{N-1}$. Using Equation 64 and Equation 61, Lagrange's constraint (Equation 56) is reformulated as:

$$\begin{aligned} L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) = & \sum_{i \in N(j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i{}^T (\mathbf{q}^{N+1}) \right) \\ & - Q_j^g - Q_j^c(\boldsymbol{\lambda}^N) - Q_j^{ext} = 0 \end{aligned} \tag{65}$$

The only terms depending on unknowns in Equation 65 are $\ddot{\mathbf{W}}_i$ and Q_j^c. All other terms can be readily evaluated using $\mathbf{q}^{N-1}$ and $\mathbf{q}^N$, which serve as constants in the

optimization. Once this optimization is solved, we advance our time by Δt, making $N+1$ as current time sample.

Our formulation has the same order of accuracy as the second-order Runge-Kutta method in solving ordinary differential equations numerically. To improve the time performance in practice, we define the velocity of a DOF q_j using backward finite differences:

$$\dot{q_j}^N \equiv \frac{q_j^N - q_j^{N-1}}{\Delta t} \tag{66}$$

This definition of joint velocity sacrifices the second-order accuracy, however, the Lagrange's equation becomes a linear function of unknowns $\mathbf{q}^{N+1}$ and $\boldsymbol{\lambda}^N$, allowing for a much more efficient quadratic programming formulation.

Our optimization framework deals with constraints that, in general, are non-linear in $\mathbf{q}^{N+1}$ and $\boldsymbol{\lambda}^N$. However, solving a constrained non-linear optimization problem is slow, in general, as compared to solving a quadratic programming (QP) problem which consists of linear constraints and quadratic objectives. Thus, it is desirable to have as many linear constraints as possible for the solver. Any constraint $\mathbf{C} = 0$ can be used as an objective function e.g. as $\mathbf{C}^T\mathbf{C}$ or $(\|\mathbf{C}\|^2)$ to be minimized in the optimization along with other objectives. Thus, linear constraint can be used as an quadratic objective in a QP problem.

1. *Lagrange's constraint.* Equation 65 gives the discretized Lagrange's constraint as a function of $\mathbf{q}^{N+1}$ (by using Equation 64 and Equation 62) and $\boldsymbol{\lambda}^N$. From Equation 72, we see that the generalized contact force Q_j^c is linear in $\boldsymbol{\lambda}^N$. The constraint is also linear in $\ddot{\mathbf{W}}_i$'s, which are functions of $\mathbf{q}^{N+1}$. Now if we use the definition of joint velocity as in Equation 61, Lagrange's constraint becomes quadratic in $\mathbf{q}^{N+1}$. However, if we use the definition in Equation 66, the constraint becomes linear in $\mathbf{q}^{N+1}$. This motivates us to sacrifice the second order accuracy for a practical speedup in solving an optimization. Note that this makes the corresponding objective (as used in Section 3.4.1) quadratic.

2. *Position constraint.* A position constraint $\mathbf{C}_P$ which fixes the position of some point $\mathbf{p}_l$ on body node i to a world position $\mathbf{p}_0$ i.e. $\mathbf{C}_P = \mathbf{W}_i^{N+1}\mathbf{p}_l - \mathbf{p}_0 = 0$, is non-linear in $\mathbf{q}^{N+1}$. We linearize it by approximating the position of this point by using its position at current time sample N and velocity at time sample $N+1$:

$$\begin{aligned}
\mathbf{C}_P(\mathbf{q}^{N+1}) &= \mathbf{W}_i^{N+1}\mathbf{p}_l - \mathbf{p}_0 \\
&\approx \mathbf{W}_i^{N}\mathbf{p}_l + \dot{\mathbf{W}}_i^{N+1}\mathbf{p}_l\Delta t - \mathbf{p}_0 \\
&= \mathbf{W}_i^{N}\mathbf{p}_l + \frac{\partial \mathbf{W}_i^{N+1}}{\partial \mathbf{q}^{N+1}}\dot{\mathbf{q}}^{N+1}\mathbf{p}_l\Delta t - \mathbf{p}_0 \\
&\approx \mathbf{W}_i^{N}\mathbf{p}_l + \frac{\partial \mathbf{W}_i^{N}}{\partial \mathbf{q}^{N}}\dot{\mathbf{q}}^{N+1}\mathbf{p}_l\Delta t - \mathbf{p}_0 \qquad (67)
\end{aligned}$$

Similarly other functions of positions e.g. COM position constraint, which can be computed as a linear combination of COMs of individual body nodes, can be approximated in this fashion.

3.4.1 Muscle control

With only 6 equations (Equation 65) on the root DOFs, this system is largely underdetermined and has infinitely many solutions. We do not enforce Equation 65 on joint DOFs as they are implicitly equipped with muscles or actuators that can generate arbitrary forces to satisfy Equation 65. This formulation is equivalent to computing the aggregate force and torque [29] and the low-order dynamic constraints [85].

To bias the solution towards a more plausible configuration, we incorporate the minimal torque change model in the optimization [43, 88]. Natural human motion tends to remain smooth in the acceleration domain with limited ability to change the muscle activation rapidly over time.

Therefore, minimizing the change of joint torques in time discourages the muscle forces from changing abruptly and excessively.

We define the generalized muscle force usage at each actuated joint DOF q_j as Q_j^m

, which represents the sum of torques generated internally by musculoskeletal components. The Lagrange's constraint for each actuated DOF can then be expressed as:

$$L_j{}'(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) = L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - Q_j^m = 0 \tag{68}$$

Using this equation, change of muscle force is defined as:

$$\dot{Q}_j{}^m = \dot{L}_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) \equiv L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - L_j(\mathbf{q}^N, \boldsymbol{\lambda}^{N-1}) \tag{69}$$

where $L_j(\mathbf{q}^N, \boldsymbol{\lambda}^{N-1})$ is the muscle force at previous time step and serves as constant in the equation.

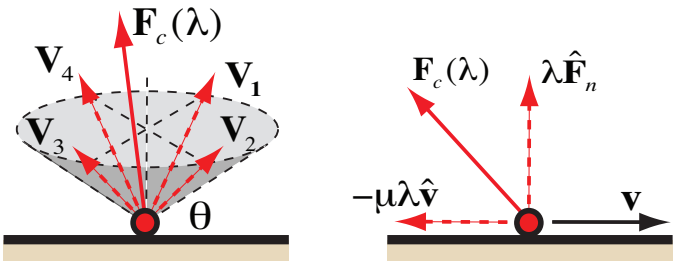
We propose a simple method to regulate the muscle forces to achieve natural human motion. We add objectives, $\dot{L}_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N)$, for each actuated DOF q_j, to minimize the change in muscle forces over time. When the character reacts to unexpected events, such as being pushed by the user, we simulate the activation delay in adjusting the muscles by minimizing the objective $L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) - L_j(\mathbf{q}^{N_0}, \boldsymbol{\lambda}^{N_0-1})$ for a small time interval ($\approx$ 200ms), where $L_j(\mathbf{q}^{N_0}, \boldsymbol{\lambda}^{N_0-1})$ is the muscle force usage at the moment of the push. This delay in muscle response is due to the delay in internal spinal feedback loop and external visual feedback loop [43, 61] Such simple feature results in a natural passive reaction to the push.

3.4.2 Contact model

Our method explicitly optimizes the contact forces subject to Lagrange's equations of motion and the Coulomb's friction model. This implies that the character can use any contact forces, within correct range of the friction model, to satisfy Equation 56 and help achieve other objectives.

The standard optimization formulation usually handles a sustained contact by adding a positional constraint and Lagrangian multipliers parameterizing the contact force in the dynamic equations. The drawback of this setup is that, instead of breaking off the contact, the optimization will become infeasible when the equations of motion

cannot be satisfied simultaneously with the constraints imposed by the friction model. We instead enforce a more relaxed non-penetrating constraint that prevents inter-penetration of the points in contact but allows for contact slippage and breakage. The formulation of the contact forces depends on whether the contact is static or dynamic.



(a) static (b) dynamic

Figure 5: Contact force parametrization for different types of contacts

Static: A static contact has zero tangential velocity along the surface. According to Coulomb's friction model, the repulsive contact forces should lie within the *cone* defined by the static friction coefficient μ whose generatrix forms an angle $\theta = cot^{-1}\mu$ with the surface of contact. We approximate this friction cone by four basis vectors with non-negative basis coefficients (Figure 5a). The contact force is computed as a linear combination of these bases $\mathbf{V}$ as:

$$\mathbf{F}_c(\boldsymbol{\lambda}) = \mathbf{V}\boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq \mathbf{0} \tag{70}$$

where $\boldsymbol{\lambda}$ represents the coefficient vector $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$.

Dynamic: A dynamic contact slips along the surface with the friction force directed opposite to the velocity, $\mathbf{v}$, of the contact point (Figure 5b). The contact force is computed as

$$\mathbf{F}_c(\boldsymbol{\lambda}) = (\hat{\mathbf{F}}_n - \mu\hat{\mathbf{v}})\boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq \mathbf{0} \tag{71}$$

where $\boldsymbol{\lambda}$ is a vector of one coefficient representing the magnitude of the normal contact force $\mathbf{F}_n$.

The generalized contact force, Q_j^c, for q_j, is given by:

$$Q_j^c(\boldsymbol{\lambda}) = \Big(\frac{\partial \mathbf{W}_b}{\partial q_j}\mathbf{p}_b\Big).\mathbf{F}_c(\boldsymbol{\lambda}) \tag{72}$$

where $\mathbf{p}_b$ is the contact point in the local coordinates of body node b. Depending on the type of contact, $\mathbf{F}_c$ is parameterized by $\boldsymbol{\lambda}$ consists of either one or four coefficients.

When a contact k is established, we add a non-penetrating inequality constraint, $\mathbf{C}_{np}^k(\mathbf{q}^{N+1}) > \mathbf{0}$, to the optimization, and solve for the contact forces that help satisfy the constraint. In addition, these contact forces are constrained either by static or dynamic friction forces based on the tangential velocity at the contact point. If the normal velocity is nonzero, the contact breakage occurs and we simply remove the contact from optimization for the next time step.

Apart from these contacts based on unilateral friction constraint, the character can be in contact with an object by grabbing onto it. In such a case, the contact forces are unconstrained due to the bilateral grip and we simply add three unconstrained coefficients for the forces in the optimization.

Because we only enforce a non-penetrating constraint on the contact, the character might choose to use her own muscles to unnecessarily slide along the surface or even break off the contact. Therefore we need an incentive for the character to move the contact point only when it is physically impossible to maintain the contact, or when an overpowering conflicting objective occurs. To this end, we add an objective $G_c^k(\mathbf{q}^{N+1})$ to minimize the movement of the contact point. By using these objectives to reduce the voluntary slippage and breakage, the character behaves in a more human-like manner without sacrificing physical correctness.

One drawback of this contact model is that it does not enforce the principle of zero virtual work for contact forces. That is, the contact force is not guaranteed to be zero at the moment of the breakage, resulting in a nonzero amount of work is done. When this situation is detected, we rollback our motion to the previous frame and enforce the contact force to be zero.

3.4.3 Optimization summary

Equation 73 summarizes the formulation of the optimization problem at each time step using the constraint and objective notations introduced in this section.

$$\begin{aligned} \underset{\mathbf{q}^{N+1},\boldsymbol{\lambda}^N}{\operatorname{argmin}} \quad & E = \sum_{j=6}^{36} \|\dot{L}_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N)\| + \sum_k \|G_c^k(\mathbf{q}^{N+1})\| \\ \text{subject to} \quad & \begin{cases} L_j(\mathbf{q}^{N+1}, \boldsymbol{\lambda}^N) = 0, \ j = 0, \cdots 5 \\ \mathbf{C}_{np}^k(\mathbf{q}^{N+1}) > \mathbf{0}, \ \forall k \end{cases} \end{aligned} \tag{73}$$

3.5 Framework for active control

The motion synthesizer described in Section 3.4 produces physically plausible motion with regards to frictional contacts and smooth changes in muscle activations. Without any active control, however, the character will quickly fall on the ground under the influence of gravity. The goal of the controller is to direct the virtual character's active motion in reaction to the environment to achieve a specified task.

In our framework, the controller comprises a user-specified control strategy that maps the character's dynamic state and the environment state to an appropriate set of objectives and constraints. These objectives and constraints describe the desired goal of an active motion as a function of the character's joint position and derivatives. At each time step, the controller determines appropriate control strategies and adds the desired objectives and constraints to the current optimization problem. Consequently, the optimizer must generate a motion that follows the dictates of the controller while also satisfying the physical constraints in the environment.

Our framework allows for a more intuitive specification of controller behavior than prior optimal control algorithms for dynamic system. In prior optimal control algorithms, a controller is a model of the physical actuators responsible for generating the internal force that creates a desired motion. In our framework, a controller is essentially a statement of the kinematic goal of the motion, specified as functions of joint

DOFs $\mathbf{q}^{N+1}$ and the external contact forces $\boldsymbol{\lambda}^N$. To design a controller for complex interaction with the environment, our approach allows the programmer to intuitively describe control strategies as a sequence of kinematic actions, such as desired poses and potential contacts with the environment. The entire process of controller design does not require the programmer to fine tune the physical parameters representing the joint actuators. However, to create a specific output motion, the programmer has to find a balance set of weights for the objectives. In our experience, tuning objective weights is relatively easier because the weights only determine the high-level relative importance among competing objectives, rather than the physical properties of joints and muscles. Consequently, one set of weights is consistently applied across all joints and can be used for different characters in different environments. Furthermore, a wide range of weights can produce different but equally plausible motion sequences. We provide the exact weight settings used in our examples in Section 3.6, but the programmer can vary these values to create a variety of motions.

3.5.1 Controller specification

Formally we define a controller as a finite-state machine (FSM) $\mathbf{M} = (\mathbf{S}, \mathbf{T})$ with states $\mathbf{S}$ and allowed transitions $\mathbf{T}$. Each state is a basic control strategy comprising a set of control objectives $(G_1, G_2, \ldots)$ representing the kinematic goals of the desired motion, and a boolean condition D representing when the strategy is applicable (a condition returns **false** when it fails or is not applicable). Control objectives, G_i's, can be represented as functions of the character's joint positions: $G_i = \|f(\mathbf{q}^{N+1})\|$. Each objective function has an associated weight which indicates its relative importance in the optimization. We assign these weights based on our understanding of human locomotion and experimentation. In our experiments, we never needed to scale the weights for different joints. Once these values are tuned, we do not need to change them for different characters or environments.

Transitions determine allowable changes in control strategy within a single simulation time step. If the current control strategy is not applicable, then the state machine searches for an adjacent control strategy that is reachable through a transition. Our formulation is inspired by an approach described by Faloutsos et al. [28]. The main difference is that in Faloutsos's work, a state has both a precondition for entering and postcondition for leaving. In all our examples, a single condition suffices for the behaviors we wished to implement. At each simulation time step, the FSM searches for a control state in $\mathbf{S}$ that is applicable to the current dynamic state of the character and the environment state described by the condition D associated with each control state. The motion synthesizer then adds the state objectives to the current optimization.

To demonstrate the ease of controller design using our API, we show the implementation of a balance controller, a climb controller and a swing controller in the next section.

3.5.2 Environment knowledge

Realistic virtual characters incorporate sensor information about their surrounding environment to determine the appropriate action according to the desired task. We demonstrate the capability of our framework to model this behavior for a simplified synthetic sensory system. We endow our character with a sensor that can evaluate the reachability of environment objects and surfaces with respect to the character. For example, the character can regain balance using anything she can reach and grab onto in her immediate surroundings, such as handles, poles, or walls. In terms of the control algorithm, this entails augmenting the environment state with a list of objects that are reachable by the virtual character's end effectors. We can then specify control strategies where the condition incorporates information about the reachability of particular objects, and the objectives can describe desired spatial or

derivative relationships between the character and object.

3.6 Implementation and results

We now discuss the design of several controllers that enable the character to perform various actions in a varying environment.

The motion for all the examples discussed in this section is simulated at 2-10 frames per second on a single core of 2.93 GHz Intel Core 2 Duo processor. The variance in simulation time is primarily due to the complexity of the controllers. Full animations can be seen in the supplemental video available online in the ACM digital library [41]. We used SNOPT [33] to solve the optimization problem at each time step. The time step used for simulation is 0.01 seconds. Our framework does not require any motion capture sequences.

3.6.1 Balance controller

Balancing is the basis for all locomotion tasks for bipedal character. In this section, we describe the implementation of a balance controller using the controller specification described in Section 3.5.1. We start by describing a basic balancing strategy that allows the character to stand on the ground and maintain balance. We then enhance this basic control strategy with complex ones that allow the character to take protective steps when required or utilize nearby surfaces to recover balance. The same balance controller can be applied to different behaviors (dodging incoming objects, standing one one foot), different physical models (child character), and different environments (balancing on the ice) by adding a few high level objectives.

3.6.1.1 Basic Balance Strategy

We implemented a basic balance strategy with the following high level objectives:

1. Support the COM, $G_{cp} = \|proj(\mathbf{COM}(\mathbf{q}^{N+1})) - \mathbf{C}_{sp}(\mathbf{q}^{N+1})\|$, where **COM** is the center of mass, $proj()$ projects a point to the ground and $\mathbf{C}_{sp}$ is the center

of support polygon evaluated at time sample $N+1$.

2. Keep upper body upright, $G_{spine} = \|\mathbf{d}_{spine}(\mathbf{q}^{N+1}) \times \hat{j}\|$, where $\mathbf{d}_{spine}$ is spine orientation at time sample $N+1$ and $\hat{j}$ is the direction of gravity.

3. Avoid sudden movements, $G_{qv} = \|\dot{\mathbf{q}}^{N+1}\|$

We create a state representing the balance strategy, *balance*, with the weighted sum of objectives mentioned above:

balance Objective: $5.0G_{cp} + 70.0G_{spine} + 0.5G_{qv}$
balance Condition: **if** COM is outside the support polygon, **return false**; **else return true**

This state machine comprising of only one state with three simple strategies is capable of maintaining a balanced pose for the character even under small perturbations (see supplemental video).

The exact same balance strategies also allow the character to balance on one foot. By reducing the supporting polygon to an arbitrarily chosen supporting foot, the character automatically shifts her weight toward the supporting foot. Once the character balances herself with one foot, we add kinematic objectives to make the character mimic one given pose (Figure 9a).

3.6.1.2 Enhanced Balance Strategies

The *balance* state fails when the COM of the character falls outside the support polygon. To handle this failure, we add two states, *relaxFoot* and *takeStep*, to the basic balance controller that enable the character to take protective steps, by automatically deciding when and where to place the foot for recovering balance. The more robust balance controller is represented below:

BALANCE Machine:

States:

$balance$, $relaxFoot$, $takeStep$

Transitions:

$balance \rightarrow relaxFoot$

$relaxFoot \rightarrow takeStep$

$takeStep \rightarrow balance$

relaxFoot. The $relaxFoot$ state is responsible for reducing the ground contact forces on the foot about to be lifted before taking a step. The decision of which foot is to be lifted depends on a simple heuristic that assumes the foot farther from the ground projection of the COM is easier to lift. To relax the forces on the foot, we add the following objectives:

1. Move COM to the supporting foot, G_{cf}.
2. Relax contact forces on the foot to be lifted, $G_c = \|\boldsymbol{\lambda}^N\|$, where $\boldsymbol{\lambda}^N$ are the contact force parameters for the contact points on the foot.

This helps the character to shift her weight away from the foot to be lifted and eventually reduce the contact forces. The rationale behind moving the COM towards the supporting foot comes from our observations of recorded human motion in which the COM accelerates towards the supporting foot before the subject breaks the contact from the other foot.

$relaxfoot$ Objective: $0.2G_{qv} + 50.0G_{cf} + 1.0G_c$.
$relaxfoot$ Condition: **if** contact forces on foot are relaxed, **return false**; **else return true**

takeStep. In the *takeStep* state, we have an objective to move the lifted foot to the desired position. The desired foot position is updated at each time step when the character is in this state. It is based on the simple heuristic that the COM should lie in the center of support polygon. Thus, the new foot position, $\mathbf{p}_f$, is chosen such that ground projection of COM lies midway between the feet. The objective G_p for moving a body point, $\mathbf{p}_i$, defined in local coordinates of body node i, to any desired position $\mathbf{p}_0$ is written as $G_p = \|\mathbf{W}_i^{N+1}\mathbf{p}_i - \mathbf{p}_0\|$. Thus, we substitute the desired position $\mathbf{p}_0$ in this equation by $\mathbf{p}_f$.

takeStep Objective: $3.0G_{qv} + 20.0G_{spine} + 0.8G_p$

takeStep Condition: **if** distance between the moving foot and the desired position is increasing, **return false**; **else return true**

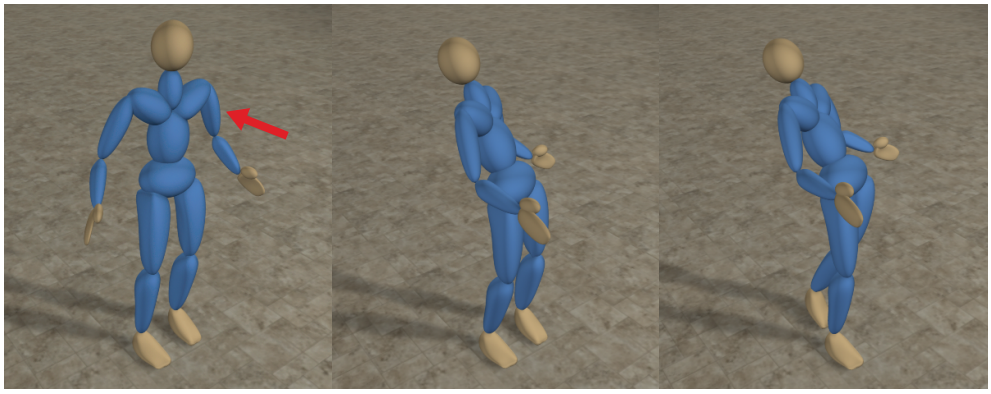
At each time step, progress of moving foot is monitored and when the above condition fails i.e. situation is getting worse as the character is not able to move her foot as fast as it should, the state machine decides to place the foot on the ground and transition to the *balance* state once again.

This completes one protective step to recover balance and if the *balance* fails again, this cycle is repeated. e.g. in case of a strong push, the character has to take multiple steps to recover (see Figure 6).

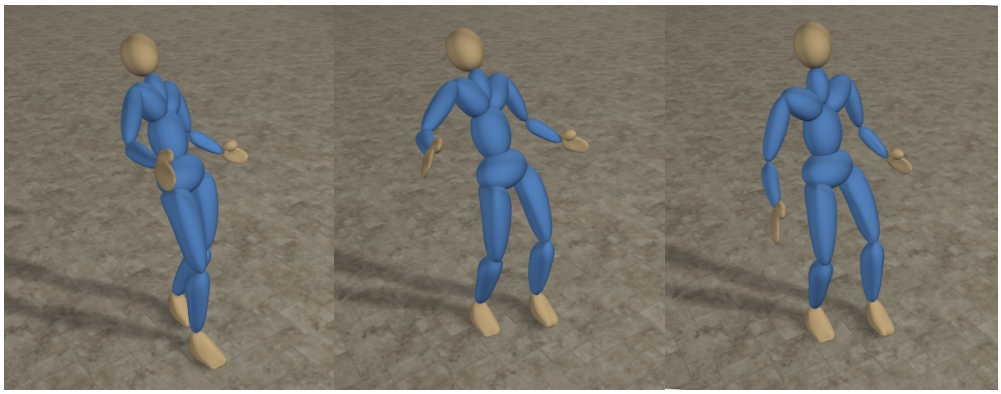
To demonstrate the robustness of the balance controller, we just change the friction coefficient of the floor from 1.0 to 0.2 to model icy surface. When the character tries to recover from a push on a slippery surface, she often slips and cautiously takes smaller steps to reduce this slipping (see Figure 7).

3.6.1.3 Support using Environment Features

By incorporating the balance controller with information from a synthetic visual sensory system, we develop a balance controller that synthesizes significantly different



(a) pushed backwards (b) trying to balance (c) taking step



(d) still not balanced (e) taking additional step (f) finally balanced

Figure 6: Character going through a series of states of the balance controller after she is pushed and finally balances after taking a couple of steps (red arrow depicts the applied force).

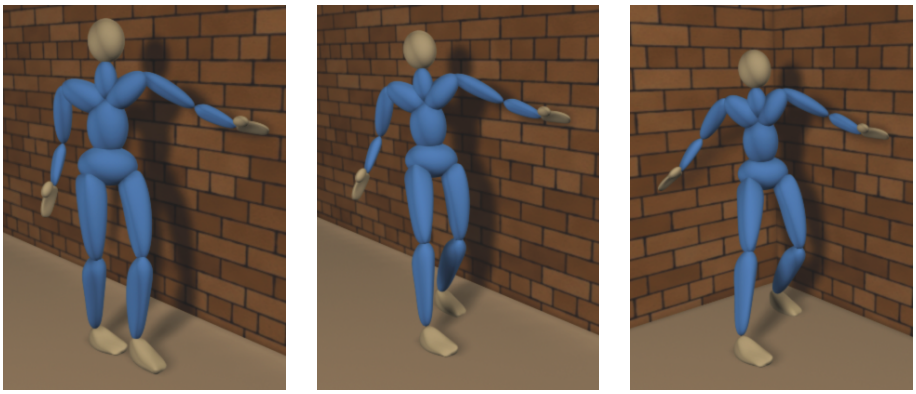
(a) pushed backwards (b) slipping while trying to balance (c) taking small steps

Figure 7: Character trying to balance on an icy surface when pushed by taking small steps and slipping occasionally

motions in response to different environmental stimuli.

The visual sensory component takes as input the character's current configuration and the current state of all objects in the environment, and outputs a list of objects reachable by any end-effector on the character's body. We define an end effector, e_i, as a point on the body that can be used for support (e.g. a hand or a foot) against a reachable object, o_j. For each $(e_i,\ o_j)$ pair, the character evaluates whether o_j is reachable by e_i. If so, we add an objective in the motion synthesizer that moves e_i towards o_j.

We demonstrate that the character autonomously determines to use the nearby wall for support when pushed by a large force. The character automatically decides when to move her hand for support and tries to reach for the nearest point on the wall (projection of her hand on the wall). By increasing the distance between the wall and the character, she takes extra steps before reaching the wall. The reaction changes significantly when there are two walls available for support (Figure 8). When there are multiple available contact points, the character simply reaches out for the closest one. Nonetheless, more sophisticated strategies can be encoded in the controller. This capability of using environment features for support is added on top of the same balance controller as described earlier. We refer the reader to the supplemental video for full animations.



(a) nearer wall (b) further wall (c) two nearby walls

Figure 8: Character's reaction to the same push in different environment settings

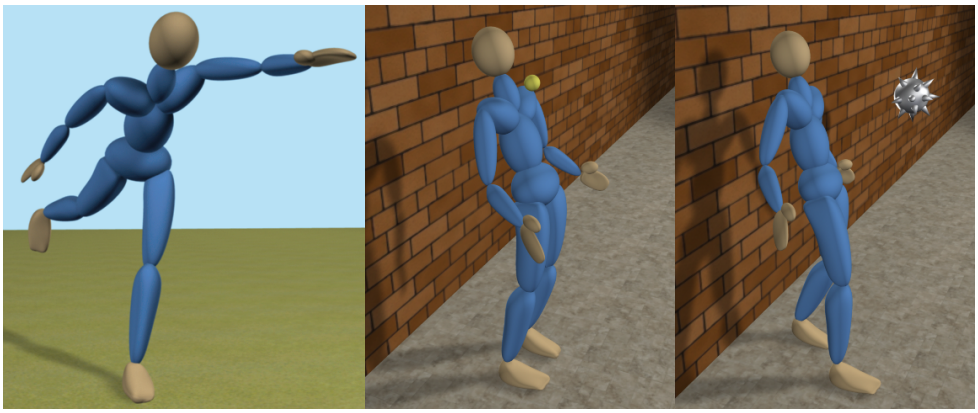
3.6.1.4 Balance and Dodge

To create more interesting behavior in a dynamically varying environment, the programmer can add simple high level objectives to the balance controller. For example, we create a behavior where a character dodges the objects thrown at her while maintaining balance. We add a dodging objective that keeps the character's body parts away from the object. For an object at position $\mathbf{p}$ and traveling with velocity $\mathbf{v}$, we add objectives to maximize (by putting negative weight for objective) distance of some points p_i (defined in local frame of body node i) on the body which lie near to the line $\mathbf{l}$ passing through $\mathbf{p}$ with direction $\mathbf{v}$ (parametric representation $\mathbf{l}(t) = \mathbf{p}+\mathbf{v}t$). Thus, the objective can be written as $G_{dodge} = dist(\mathbf{W}_i^{N+1}\mathbf{p}_i, \mathbf{l})$, where $dist()$ evaluates the distance of a point $\mathbf{p}_i$, when expressed in world coordinates, to the line $\mathbf{l}$. We weight the dodge objectives as inversely proportional to the distance to the line (-0.05 to -0.15) and are active when the object is within a certain distance (e.g. 1 meter) from the character.

In addition, by adjusting the relative importance of each objective, the same controller can produce a variety of behaviors. In the synthesized example, the character easily dodges the tennis ball by bending her spine (Figure 9b) but gets hit on the arm by the object coming from behind. The programmer can adjust the importance of dodging objective based on the incoming object. If the character sees an flying object that appears harmful, she quickly moves out of way (Figure 9c). To realize this, we changed the weights of objective function in the *balance* state (Section 3.6.1.1) to $2.0G_{cp} + 30.0G_{spine} + 0.5G_{qv}$ for a harmful object.

Better dodging strategies or hazard assessment rely on domain knowledge in controller design. [107] learns different anticipation strategies from motion capture data and automatically chooses which to employ at runtime according to a damage and energy assessment calculated from simulation results. Their results exhibits a wide variety of dodging strategies with compelling realism. [65] uses psychological insights

and motion capture data to formulate protective anticipatory movement parameterized by a model of approaching object. This is combined with a physically-based dynamic response to produce animations with anticipation and reaction to impacts. We do not aim at thoroughly solving a particular problem of anticipation, rather emphasize the ease of designing control strategies with limited domain knowledge.



(a) balancing on one foot (b) dodging a less harmful object (c) dodging a more harmful object

Figure 9: Character performing variety of tasks while balancing

3.6.1.5 Multiple Character Interaction

The same balance controller can operate across characters with different mass distribution and skeletal structures. In addition, the programmer can simulate interactions between multiple characters by adding objectives or constraints that model the physical contacts. We synthesize a child reaching out for an adult's hand for support in the event of balance loss (see Figure 10). The natural reaction for both the characters is synthesized automatically, since the objective of holding hands and force exchange affects both the characters' joint configuration.

All the examples described in this section took 2-4 frames per second to simulate. The slow simulation speed is due to the complexity of the balance problem as the objectives conflict and compete with each other in the optimization.

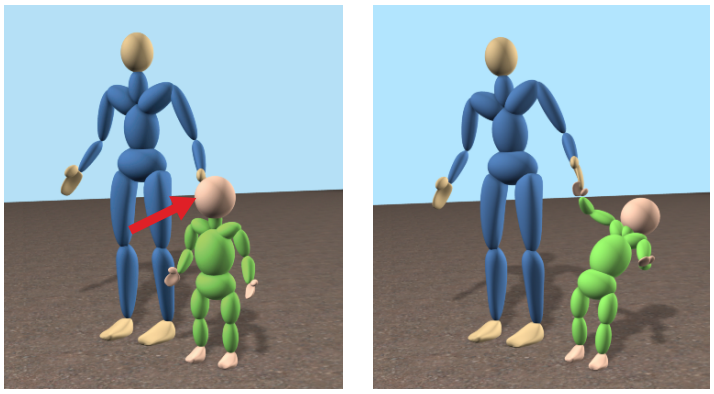


Figure 10: Adult character preventing the child character from falling by holding hands when the child character is pushed.

3.6.2 Climb controller

We next move to a control task that requires a much more sophisticated interplay between the character and environment. We implement a climbing controller that facilitates wall climbing using attached holds. The holds are placed at random positions and the character automatically decides which ones to grab in order to progress upwards. A complex wall climbing motion can be generated by specifying kinematic constraints at the hand holds and foot holds.

We create five states $allSupport$, $relaxHand$, $moveHand$, $relaxFoot$, $moveFoot$ and define the state machine for the climb controller as follows:

$CLIMB$ Machine:

States:

$allSupport$, $relaxHand$, $moveHand$, $relaxFoot$, $moveFoot$

Transitions:

$allSupport \rightarrow relaxHand$

$relaxHand \rightarrow moveHand$

$moveHand \rightarrow allSupport$

$allSupport \rightarrow relaxFoot$

$relaxFoot \rightarrow moveFoot$

$moveFoot \rightarrow allSupport$

allSupport. In this state, the character grabs both the hand holds and places her feet on the foot holds. This state consists of following objectives:

1. To raise her COM to the highest possible position, so that she is comfortable to stretch out her hand and grab the next hand hold. The objective for the COM can be written as $G_{com} = \|\mathbf{COM}(\mathbf{q}^{N+1}) - \mathbf{C}_0\|$, where $\mathbf{C}_0$ is the center of hand holds that is high enough for the COM to reach.

2. To reduce the joint velocities for smooth movements, G_{qv}.

allSupport Objective: $0.2G_{qv} + 20.0G_{com}$
allSupport Condition: **if** the character is stable, **return false**; **else return true**

relaxHand. The goal of the character is to relax the contact forces on the hand so that it can release the hold and move to the desired position. We achieve this by setting objectives for reducing the contact forces from the corresponding hand hold (*moveFoot* has similar objectives for relaxing the contact forces on the foot).

relaxHand Objective: $0.2G_{qv} + 1.0G_c$
relaxHand Condition: **if** the contact forces fall below a small threshold, **return false**; **else return true**

moveHand. In this state, the controller adds the position of next nearest hand hold, along with minimization of joint velocities as control objectives (see Figure 11b).

moveHand Objective: $0.2G_{qv} + (0.1 \text{ to } 0.25)G_p + 10.0G_{com}$

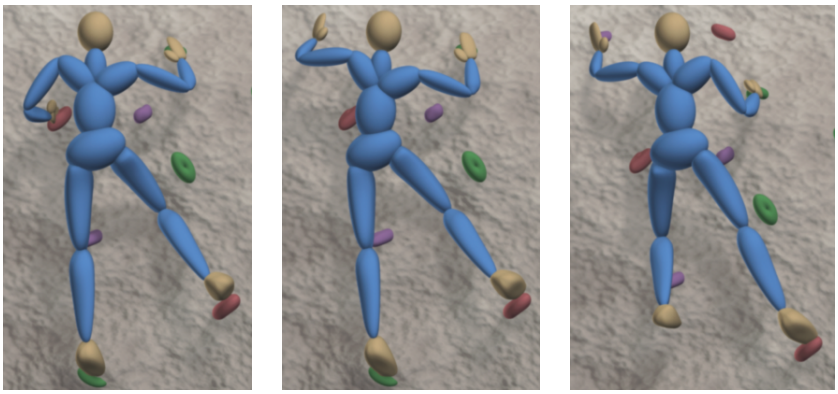
moveHand Condition: **if** hand reaches the hand hold, **return false**; **else return true**

We vary the weights for G_p according to the distance of the hand to the desired position (higher weight when nearer). When the condition fails, i.e. hand reaches the hand hold, grasping contact is established and state transitions to *allSupport*. Next, to raise her body up and move her foot, the character relaxes the forces on her foot by making a transition to *relaxFoot* state. Once relaxed, she moves to *moveFoot* state.

moveFoot. In this state, the position of next foot hold is set as an objective for the moving foot and the character begins to move her foot to the desired position (see Figure 11c). When the foot reaches close to the hold, we add an objective, $G_{pose} = \|\mathbf{q}^{N+1} - \mathbf{q}_0\|$, to guide the joint angles close to the starting pose, $\mathbf{q}_0$, to make the character assume a realistic looking pose. This starting pose (see Figure 11a) is created using simple inverse kinematics.

moveFoot Objective: $0.2G_{qv} + (0.01 \text{ to } 0.30)G_p + 10.0G_{com} + 2.0G_{pose}$

moveFoot Condition: **if** foot reaches the foot hold, **return false**; **else return true**



(a) *allSupport* (b) *moveHand* (c) *moveFoot*

Figure 11: Character in different states of climb controller

This completes one cycle of the state machine which moves the character up by one hold and makes her reach a stance similar to the starting pose. By looping over this cycle, the character can climb arbitrary number of holds.

In the synthesized example for climbing a wall (see supplemental video), the programmer only needed to specify the starting pose for the character designed using simple inverse kinematics, and the placement of holds on the wall. With the help of simple strategies and kinematic constraints described above, the character automatically climbs up the wall using required amount of external contact forces from these randomly placed hand and foot holds. The simulation rate for this example varied from 5-10 frames per second.

3.6.3 Swing controller

Our framework facilitates synthesis of natural motion by defining a few high level objectives. Thus, it can serve as a test-bed for designing new motor skills. In this section, we describe a simple swing controller, based on only one objective function, maximize the center of mass velocity in the direction tangential to her movement. The character starts from a rest pose holding a high bar with both hands. We create a simple state machine $SWING$ consisting of two states $trySwing$ and $passiveSwing$.

In $trySwing$ state, the character tries hard to increase her COM velocity in the direction perpendicular to the plane joining her COM and grasps on the bar. The objective for increasing the COM velocity is defined as $G_{cv} = \|\mathbf{C\dot{O}M}(\mathbf{q}^{N+1}) - \mathbf{v}_d\|$, where $\mathbf{v}_d$ is the desired velocity (a value more than the current COM velocity).

$trySwing$ Objective: $0.1G_{qv} + 4.0G_{cv}$

$trySwing$ Condition: **if** the angle of swing increases a threshold, **return false**; **else return true**

When the angle of swing increases above a threshold, a transition to $passiveSwing$

occurs. This state's objective is to maintain constant velocity in the perpendicular direction to create a smooth passive swing.

Based on only one objective function and no knowledge in gymnastics, the character tries hard to increase her velocity and is able to start swinging. However, she is not able to increase her angle of swing beyond a certain limit because of the lack of coordination of her joints and skills possessed by gymnasts (see Figure 12a). When started from a higher angle, the character still fails to maintain the momentum and the velocity diminishes rapidly

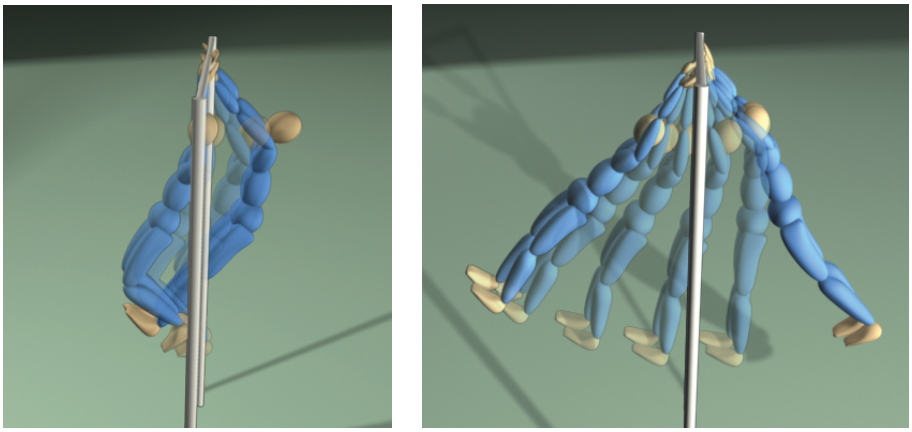
Adding some more objectives to improve the coordination of joints help the character maintain her velocity. To achieve this, we added joint position objectives to make her body more stiff and legs straightened, then let her start from a higher swing angle. These objectives are meant to keep her legs close to a specific pose (straight legs) and the stiffness is achieved by setting a relatively higher weight for these objectives.

The objective function now becomes $0.1G_{qv} + 15.0G_{cv} + 5.0G_{pose}$. G_{pose} is responsible for stiffening the body and straightening the legs. We do not add pose objectives for DOFs of the abdomen to allow easier bending of the abdomen.

The character is now able to swing more smoothly and is able to maintain her velocity (see Figure 12b). The examples were synthesized at 5-10 frames per second.

3.6.4 Composition of multiple controllers

Primitive controllers can be easily composed to create an autonomous and versatile virtual character. The cable car example (Figure 13) highlights realistic behaviors and responsive reaction of the character to unexpected events in a dynamically varying environment. Initially, the character comfortably counteracts small disturbances of the cable car with her balance strategies. When the car shakes violently, she decides



(a) start from rest position (b) start from high angle

Figure 12: Character swinging with different initial swing angle

to take protective steps and grab onto nearby walls and bars to prevent herself from falling. When the ground breaks, she resorts to holding the bar and apply her swing motor skills to hang on (see supplemental video). The programmer just key-framed the events like rocking the cable car and breakage of walls and the floor, and the character autonomously decides what and when to grab, depending on her immediate surroundings and her dynamic configuration.

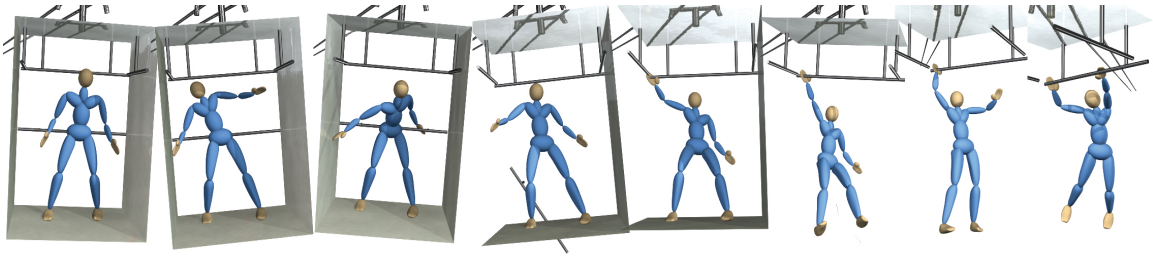


Figure 13: An autonomous character reacting to unexpected events inside a cable car and trying hard to prevent herself from falling

3.7 Discussion

The design of our approach raises some important issues and questions in practice. First, to what degree of physical realism can the system provide when the user-specified objectives are conflicting or unrealistic? Second, is the weight adjustment

in the objective function any easier than parameter tuning for designing robotic controllers? Third, what types of motions/tasks are mostly appropriate to our framework?

3.7.1 Physical realism

We enforce exact equations of motion on the global dynamics of the character i.e. at the root of character's hierarchy where no actuators exist. These equations (Equation 65) alongside the contact model (Section 3.4.2) are physically correct up to discretization error that is similar to numerical integration methods used for forward simulation.

For all the other joints, we do not enforce these equations implying that the actuators at these joints can assume arbitrary values. However, we add an additional objective for minimizing the change in joint torques (see Section 3.4.1), that restricts arbitrary changes in joint torques leading to smooth and plausible muscle forces. We chose not to enforce explicit joint torque limits (constraints) in the optimization because, in practice, the values of these torques remain in reasonable limits. Therefore, removing these constraints help reduce the computation time without affecting the output motion.

The ratio of the weight of muscle minimization to the weight of kinematic objectives specified by the programmer indicates the responsiveness of the character to the control goals. The user can adjust the ratio to explore the tradeoff between the naturalness of the movement and the satisfaction of control goals. However, the global physical realism is ensured regardless of the value of this ratio.

3.7.2 Weight adjustment

Our framework facilitates easy design of control strategies for articulated characters (see Section 5). The parameters required to be tuned in our framework are the weights associated with each objective function used in the optimization. These

objectives indicate high level behaviors that can be described by the joint angles such as basic balance strategies or reaching for objects. Consequently, the user can express controllers at the kinematic level without knowing the mechanical details of each joint. For example, the user only needs to tune two weights to maintain a supported center of mass and to achieve a specific location for an end-effector. In other methods such as robotic controller framework, the physical parameters for each joint need to be individually tuned and tested. Often time the values of those parameters cannot be easily translated to high level task description. Moreover, when the physical properties of the articulated body system change, readjusting physical parameters is likely to be required. In our framework, however, the same weights can be reused as long as the relative importance of the objectives remains the same.

3.7.3 Suitable range of tasks

In principle, with careful design and sufficient engineering effort, most tasks demonstrated by our framework can also be achieved by robotics controllers. However, we believe our method significantly reduces the engineering effort for the following types of tasks:

1. *Tasks that require precise positional control:* For example, hand reaching out for a moving point in space. Our method directly imposes positional control as objectives or hard constraints, rather than employing additional inverse kinematics and inverse dynamics computation to obtain the required joint torques. This type of control was frequently applied in our examples, such as step taking in the balance controller (Section 3.6.1.2).

2. *Tasks that are highly constrained by the environment through resting contacts:* In general, having more resting contacts complicates the computation in a dynamic system. Our approach, in contrary, works particularly well with multiple

resting contact. This is because contacts introduce additional degrees of freedom, contact forces, in the optimization, that "help" the character meet the various objectives with ease. Furthermore, contacts provide additional kinematic hints for solving an under-determined joint configuration.

3. *Multi-objective tasks with conflicting objectives:* Our optimization method resolves the trade-offs between conflicting objectives simultaneously with other dynamic and kinematic constraints imposed on the character. This flexible framework allows the programmer to compose simple tasks in the position domain to create complex behaviors. For example, dodge and balance tasks (Section 6.1.4) have conflicting preferred joint configurations. The programmer only needs to tune the weights of these two objectives to arrive at a solution satisfying both. In our experience, there is a wide range of weights that achieve the goal.

There are certain situations where our framework does not offer many advantages and use of other approaches might be more appropriate:

1. Existing forward simulation methods score over our approach in two situations. First, when the character does not exhibit active control in the motion (e.g. ragdoll), our optimization formulation adds unnecessary computation to a relatively trivial simulation problem. Second, when the motion involves frequent passive colliding contacts (e.g. falling off the stairs), our approach becomes very inefficient because each contact increases the number of constraints and expands the dimension of degrees of the freedom in the optimization.

2. We sacrifice the anticipatory property of spacetime optimization for interactivity. Our system can only generate anticipatory motion enforced by kinematic constraints, such as changing the kinematic goals gradually, but is not able to

create natural anticipation and follow-through involving a change of dynamics, such as a broad jump.

CHAPTER IV

LONG-TERM CONTROL PLANNING IN MODAL SPACE

In the previous chapter, we did not use any reference motion for synthesizing novel motions for the character. In this chapter, we present a control algorithm for simulating an articulated character performing a given reference motion and its variations [40]. The unique feature of our controller is its ability to make long-horizon plan at every time step. Our algorithm overcomes the computational hurdle by applying modal analysis on a time-varying linear dynamic system. We exploit the properties of modal coordinates in two ways. First, we design separate control strategies for dynamically decoupled modes. Second, our controller only applies long-horizon planning on a subset of modes, largely reducing the size of the control problem. With this decoupled and reduced control system, the character is able to execute the reference motion while reacting to unexpected perturbations and anticipating changes in the environment. We demonstrate our results by simulating a variety of reference motions, such as walking, squatting, jumping, and swinging.

4.1 Introduction

The ability to respond to the changes in the environment and predict the consequences of our own action is fundamental for everyday motor tasks. Physically simulating a virtual character who exhibits both reactive and anticipatory behaviors presents immense challenges in many facets. First, the human motor system is both underactuated and redundant. The former leads to complex issues with balance while the latter results in a high-dimensional and under-constrained problem. Second, the interaction with the environment via contacts is discrete in nature. The discontinuity introduced by the change of contact states further complicates both simulation and

control problems.

One possible approach to achieving both reactive and anticipatory virtual character is to formulate a long-term planning problem, such as spacetime optimization, and update the plan at every time step according to the current state of the character and of the environment. Motion produced by long-term planning usually appears more compliant because the character is not always in the urgency of matching the immediate goal. In addition, the frequent replanning allows the character to respond to unexpected perturbations in a timely manner. Though straightforward, this problem is extremely difficult and prohibitively expensive to solve in practice. Long-term planning on a full human dynamic system requires us to resolve all the aforementioned challenges. To date, offline solutions to optimal trajectory problems are very sensitive to parameters and initial conditions of the problem. We certainly cannot apply such brittle solutions at every time step in an online fashion.

In this chapter, we tackle a more feasible problem: designing a control system capable of long-term planning and frequent replanning for simulating *a specific motion sequence*. We introduce a new control system that tracks the reference motion while reacting to unexpected perturbations and adapting to anticipated changes in the environment. Our key insight is that the long-term planning can be largely simplified by approximating the dynamic system using *modal analysis*. In our formulation, we do not solve one long-term planning problem in the generalize coordinates, rather, we formulate a set of control strategies in a reduced and dynamically decoupled modal coordinates. Modal analysis offers two advantages to our problem:

- **Independent control**: In the modal space, each mode is governed by an independent equation of motion. This reduces a N-dimensional optimal control problem to N independent one-dimensional problems.

- **Model reduction**: Modal analysis organizes modes by the natural frequencies

of the dynamic system. Typically a few modes are sufficient to capture the dynamic behaviors of the system. This property potentially reduces the dimension of the control variables.

In spite of these great advantages, modal analysis is only suited for linear dynamic systems. We circumvent the issue by linearizing the nonlinear dynamic equations around the current state at each time step, resulting in a time-varying linear dynamic model.

We propose a new control system that makes long-term plans based on the reference motion and revises the plan at every time step in response to perturbations in the environment. We present our results by simulating a few drastically different human motions, including walking, squatting, jumping, and swinging. The virtual character can passively respond to external forces and actively replan for new tasks. For example, we demonstrate an online modification of a normal walk motion to walk on slopes, with different step sizes and different timing of steps. We also show that anticipated changes can be achieved by modifying the reference motion on the fly, such as modification of a squatting action to pick up a heavy box, or transition from a broad jump to swing.

4.2 Related work

Synthesizing natural motion for virtual characters has been a long standing challenge in computer animation. For offline applications, physics-based trajectory optimization is able to create human-like motion that exhibits anticipatory behaviors [95, 18, 60, 72]. With some variations in formulations, these methods essentially solve for a motion sequence while minimizing a chosen objective function under physical constraints, such as the equations of motion, joint limits, or contacts. Due to high dimensionality and nonlinearity in constraints or objective function, these methods are limited to simulating simple characters [18, 60, 72] or simplified dynamics [59, 29].

Researchers have utilized motion capture data to reduce the dimensionality of the motion[75, 85], but the problem remains highly nonconvex and prone to local minima. Our method takes a different approach to simplifying the long-horizon planning problem using an approximate dynamic system. We leverage the advantages of modal coordinates such that the optimization only involves a *subset* of *decoupled* dynamic equations.

Departing from the offline approach, physics based simulation methods coupled with active control are capable of synthesizing responsive motion in an interactive setting. Much research has focused on designing controllers for performing specific tasks such as standing balance [73, 89, 79, 7, 49, 62], locomotion [38, 50, 105, 80, 90], or other complex human movements [38, 96, 28]. These controllers generate impressive results, but they usually depend on highly customized control parameters or prior knowledge of the motion. Consequently, they do not generalize well to other types of activities. Our method makes no assumption of the underlying reference motion, leading to a generic control algorithm suitable for a wide variety of motions.

Recent research work in physics-based locomotion controllers [21, 26, 68, 91, 97] improves upon the previous work in terms of robustness to changes in environment, character topology and physical properties. Intuitive interfaces to author controllers for responsive and robust locomotion tasks are presented [21, 26]. Several methods specialize in adapting walking characters on uneven or unknown terrains [68, 91, 97]. In contrast to all these methods that employ strategies specific for locomotion, our method is generic to execute different tasks. The ability to replan and change the reference trajectories online makes our method suitable to adapt to different situations.

Incorporating motion capture data with dynamic controllers has a potential to create more natural and human-like motion. A number of control algorithms have been proposed to directly track the reference mocap motion using proportional-derivative

(PD) servos and their variations [108, 104, 8, 81, 12]. With proper physical parameters for the controllers, these methods can effectively generate passive responses to external perturbations. However, many of these methods require fine tuning of physical parameters or expensive pre-computation specific to the target motion and the skeletal model. The former limits the range of the variations generated by the controller, while the latter eliminates the possibility to modify the reference motion on the fly. Our control system does not require manual tuning of physical parameters or any pre-computation, making it suitable to track reference trajectories that can be modified online. In a recent work, Lee et al. [53] demonstrated a locomotion controller that modifies the reference motion by synchronizing it with the online simulation based on the contact changes. This results in improved robustness of the tracking controller while retaining the quality of motion capture data. However, their method is based on SIMBICON-style control ([105]) for locomotion making it hard to generalize to completely new motions.

To generalize control methods for different activities, many researchers suggested exploiting optimization techniques to compute control forces based on the current state of the character. Quadratic programming is applied to regulate body center of mass [7] and momenta [62]. Multiple quadratic objectives can also be organized by prioritized optimization control [25]. Similarly, nonconvex optimization can be used to directly control kinematic goals [41]. The control forces computed from an optimization process usually result in more robust motion than simple PD tracking, but these methods still rely on short-horizon optimization, lending themselves poorly for activities that require long-term planning. In contrast, our method optimizes a window of the control forces towards a future goal, rather than meeting the immediate goal.

To circumvent the issues of short-horizon planning, some researchers explored optimal feedback control techniques which consider the entire motion trajectory for

computing the control forces. da Silva et al. [24] applied a linear quadratic regulator (LQR) to control a simplified model for maintaining dynamic balance in locomotion. The optimal control policy derived from the LQR framework minimizes the cost throughout the entire trajectory, taking into account the future. Their controller tracks a reference motion and allows some variations due to perturbations. Muico et al. [69] modified the time-varying LQR to account for the dynamic constraints violations. They developed a look-ahead control policy by constructing the ground contact force predictions. Both methods are capable of responding to small perturbations in a passive manner. However, the control forces are driven by the deviation between the current state and the reference trajectory, rather than the anticipation of the changes in the environment. To actively replan for a new task, the underlying reference motion must be modified accordingly. Unfortunately, these methods require an offline process (Ricatti Equations) to compute control parameters for each new reference motion. In contrast, our method allows online editing of the reference motion and employs a completely online process to replan a look-ahead control policy at each time step. As a result, we are able to create motion drastically different from the reference motion.

Modal analysis has been previously applied to deformable models [27, 35, 42, 14], and character animation [47]. We draw inspiration from Kry et al. [47] and develop a control algorithm that fully takes the advantage of the reduced and decoupled dynamic system in modal coordinates. Kry et al. [47] select a few modes based on heuristics and manually create motions for each selected mode. They are able to create dynamic motions for simple characters but can only synthesize kinematic motions for complex characters like dog and human since they do not use any control algorithm that can handle balance issues. We demonstrate that time-varying linearized dynamic system can be a good approximation to the dynamics of a full-body, articulated character. In our method, we control the low frequency modes by formulating a

long-term control problem at every time step, resulting in robust control algorithm. The reference trajectory for each mode is derived from the given reference motion sequence rather than being manually defined.

Much previous work has explored a variety of dimension reduction techniques for character motion synthesis. Some techniques parametrize a subspace for control based on the motion data [75, 16, 102], while others define an abstract model based on domain knowledge or heuristics about the motion and the character model [103, 68]. Safonova et al. [75] employed Principal Component Analysis (PCA) to a small set of similar example motions and selected a reduced bases to synthesize physically plausible motion in a spacetime setting. However, their algorithm still needs to solve a coupled dynamic system which requires offline computation unsuitable for interactive applications. In addition, the synthesized motion is restricted to a linear subspace of example motions. Our algorithm employs modal analysis to decouple the equations of motion. Decoupling gives a significant speedup allowing us to interactively replan for a window of time at every time step. In addition, we control a subset of modes, that depend on the physical properties of the character rather than example motions, allowing us to synthesize interactions and variations in the motion that are not present in the given reference motion. Ye and Liu [103] solved an optimal control problem based on a simplified model that abstracts the degrees of freedom (DOFs) of the character into a small number of parameters. This simplification greatly reduces the complexity of the control problem rendering it suitable to be solved for the entire motion trajectory. The computation for the optimal feedback parameters is done offline, hence the reference trajectory cannot be altered online. In other concurrent work, Mordatch et al. [68] used a spring loaded inverted pendulum model to solve for optimal control in an online fashion. The control algorithm is designed for walking or running and does not use any motion trajectory. In our method, modal analysis allows for fast computation suitable for online replanning. In addition, tracking a

reference trajectory gives us benefits of constructing a generic control strategy that is independent of the performed task and synthesizing more natural motion.

4.3 Review of Modal Analysis

Modal analysis is used to transform a multi-degree of freedom (DOF) system into decoupled single-DOF systems. We review modal analysis for linear systems [78] and discuss approximations to apply modal analysis for non-linear systems.

4.3.1 Linear multi-DOF systems

For a system with N DOFs $\mathbf{q} = (q_1, q_2, \ldots, q_N)^T$ and linear dynamics, the general equations of motion are given by:

$$M\ddot{\mathbf{q}} + D\dot{\mathbf{q}} + K\mathbf{q} = \mathbf{b} + \mathbf{f}(t) \tag{74}$$

where M, D and K are constants that denote the mass, damping and stiffness matrices respectively. $\mathbf{b}$ is some constant vector and $\mathbf{f}(t)$ denotes a time dependent generalized force being applied to the system. It is convenient to choose a proportional damping model for D i.e. $D = \alpha M + \beta K$ for some damping parameters α and β.

Now, we define a modal transformation matrix Φ whose columns, $\boldsymbol{\phi}_i$'s, are the eigenvectors of the generalized eigenvalue problem $K\boldsymbol{\phi} = \omega^2 M\boldsymbol{\phi}$ i.e. $M^{-1}K = \Phi\Omega\Phi^{-1}$, where Ω is a diagonal matrix of eigenvalues or squared natural frequencies $\Omega = \text{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_N^2)$. The columns of matrix Φ are both M and K-orthogonal i.e. we can write $\Phi^T M\Phi = M_d = \text{diag}(m_1, m_2, \ldots, m_N)$ and $\Phi^T K\Phi = K_d = \text{diag}(k_1, k_2, \ldots, k_N)$. Note that $\omega_i^2 = \frac{k_i}{m_i}$.

Using the modal transformation matrix Φ, we transform the generalized coordinates $\mathbf{q}$ to a set of *modal coordinates* $\mathbf{p}$ as: $\mathbf{q} = \Phi\mathbf{p}$. Pre-multiplying Equation 74 by Φ^T, we arrive at a new set of equations of motion that govern the modal coordinates:

$$M_d\ddot{\mathbf{p}} + (\alpha M_d + \beta K_d)\dot{\mathbf{p}} + K_d\mathbf{p} = \Phi^T\mathbf{b} + \Phi^T\mathbf{f}(t) \tag{75}$$

Because M_d and K_d are diagonal matrices, Equation 75 can be decoupled into N independent one-dimensional equations, each of which is written as:

$$m_i\ddot{p}_i + d_i\dot{p}_i + k_i p_i = b_i + f_i(t) \tag{76}$$

where $d_i = \alpha m_i + \beta k_i$ and b_i and f_i are the i^{th} elements of vectors $\Phi^T\mathbf{b}$ and $\Phi^T\mathbf{f}(t)$ respectively. For those modes with $k_i \neq 0$, called *deformation modes*, the unforced solution ($f_i = 0$) for an under-damped system ($\xi_i^2 < 1$, where damping ratio $\xi_i = \frac{d_i}{2\sqrt{k_i m_i}}$) can be written as:

$$\tilde{p}_i(t) = \frac{b_i}{k_i} + S_i e^{-\xi_i\omega_i t} sin(\omega_{d,i}t + \psi_i) \tag{77}$$

where $\omega_{d,i} = \omega_i\sqrt{1-\xi_i^2}$, and S_i and ψ_i are amplitude and phase respectively determined from the initial conditions $p_{0,i}$ and $\dot{p}_{0,i}$ by solving $p_{0,i} = S_i sin(\psi_i)$ and $\dot{p}_{0,i} = S_i(\omega_{d,i}cos(\psi_i) - \xi_i\omega_i sin(\psi_i))$. The initial conditions for all the modes, $(\mathbf{p}_0, \dot{\mathbf{p}}_0)$, can be transformed from the generalized space as $\mathbf{p}_0 = \Phi^{-1}\mathbf{q}_0$ and $\dot{\mathbf{p}}_0 = \Phi^{-1}\dot{\mathbf{q}}_0$.

If $\mathbf{I}_f$ is an impulse applied at time t_f (i.e. $\mathbf{I}_f = \int_{t_f}^{t_f+\epsilon}\mathbf{f}(t)dt$), the impulse response of the system in Equation 76 is given by:

$$\check{p}_i(t) = I_i\left(\frac{e^{-\xi_i\omega_i(t-t_f)}}{m_i\omega_{d,i}} sin(\omega_{d,i}(t-t_f))\right) \tag{78}$$

where $I_i = \int_{t_f}^{t_f+\epsilon} f_i(t)dt$ is the i^{th} element of vector $\Phi^T\mathbf{I}_f$. Adding the unforced solution (Equation 77) to the impulse response (Equation 78) gives the closed form solution for the modal state $p_i(t) = \tilde{p}_i(t) + \check{p}_i(t)$ when the impulse $\mathbf{I}_f$ is applied at time t_f.

Modes with $k_i = 0$ are called *rigid body modes*. In this case, Equation 76 reduces to $m_i\ddot{p}_i + d_i\dot{p}_i = b_i + f_i(t)$. Assuming there is no damping in rigid body modes ($d_i = 0$), the unforced solution is given by:

$$\tilde{p}_i(t) = p_{0,i} + \dot{p}_{0,i}t + \frac{b_i}{2m_i}t^2 \tag{79}$$

and the impulse response by:

$$\check{p}_i(t) = I_i\left(\frac{t-t_f}{m_i}\right) \tag{80}$$

Consolidating different modes in a common equation, we collect the coefficients of the impulse from Equation 78 and Equation 80 into a time dependent diagonal matrix $A(t - t_f)$. The modal state at any time $t > t_f$, with response to impulse $\mathbf{I}_f$ can be expressed as:

$$\mathbf{p}(t) = \tilde{\mathbf{p}}(t) + A(t - t_f)\mathbf{I}_f \quad (81)$$

$$\dot{\mathbf{p}}(t) = \dot{\tilde{\mathbf{p}}}(t) + \dot{A}(t - t_f)\mathbf{I}_f \quad (82)$$

Finally, the solution in the original space can be recovered by $\mathbf{q}(t) = \Phi\mathbf{p}(t)$ and $\dot{\mathbf{q}}(t) = \Phi\dot{\mathbf{p}}(t)$.

4.3.2 Articulated characters

The motion of an articulated character is governed by non-linear dynamic equations, to which modal analysis does not directly apply. These non-linear equations of motion for articulated characters can be written as:

$$M(\mathbf{q})\ddot{\mathbf{q}} + (C(\mathbf{q}, \dot{\mathbf{q}}) + D(\mathbf{q}))\dot{\mathbf{q}} + G(\mathbf{q}) + K\mathbf{q} + \mathbf{k}_0 = \mathbf{f}(t) \quad (83)$$

where M is the mass matrix, C is the matrix for Coriolis and centrifugal forces, G represents gravity forces and $\mathbf{f}(t)$ are the generalized forces. To model passive forces, each joint is equipped with a spring and a damper. Matrices K and D then represent the stiffness and damping coefficients of the coupled system and $\mathbf{k}_0$ is a constant vector. $\mathbf{q}$ represents the character's root position, root orientation, and joint DOFs in generalized coordinates.

Time-varying linear dynamics At any time t_0, we linearize these equations around the pose $\mathbf{q}_0$ of the character at time t_0 and zero velocity. For any deviations in the position and velocity around the state $(\mathbf{q}_0, \mathbf{0})$, $\Delta\mathbf{q} = \mathbf{q} - \mathbf{q}_0$ and $\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}} - \mathbf{0}$, we approximate the equations of motion as:

$$M(\mathbf{q}_0)\Delta\ddot{\mathbf{q}} + (C(\mathbf{q}_0, \mathbf{0}) + D(\mathbf{q}_0))\Delta\dot{\mathbf{q}} + G(\mathbf{q}_0) + K(\mathbf{q}_0 + \Delta\mathbf{q}) + \mathbf{k}_0 = \mathbf{f}(t)$$

Note that C vanishes at zero velocity. Denoting $-(G(\mathbf{q}_0) + K\mathbf{q}_0 + \mathbf{k}_0)$ by a constant $\mathbf{b}$, we rewrite the equation as:

$$M\Delta\ddot{\mathbf{q}} + D\Delta\dot{\mathbf{q}} + K\Delta\mathbf{q} = \mathbf{b} + \mathbf{f}(t) \tag{84}$$

This equation represents an approximate linear dynamics model in $\Delta\mathbf{q}$ and is similar to Equation 74. Therefore, modal analysis discussed in Section 4.3.1 can be applied to this equation as well. The continuous force function $\mathbf{f}(t)$ is broken down into a series of impulses. If we assume that the force $\mathbf{f}(t)$ remains constant over a small time step Δt, we can approximate the impulse at current time t_0 as $\mathbf{I}_0 = \mathbf{f}(t_0)\Delta t$. Based on Equation 81, the position in modal space after time step Δt is then given by:

$$\mathbf{p}(t_0 + \Delta t) \;=\; \tilde{\mathbf{p}}(t_0 + \Delta t) + A(\Delta t)\mathbf{I}_0$$

Using the modal transformation, we recover the pose at the next time step: $\mathbf{q}(t_0 + \Delta t) = \Phi\mathbf{p}(t_0 + \Delta t)$. Advancing the time by Δt , we linearize the dynamic system around the new pose and repeat the same process to compute the next modal state.

4.4 Control methodology

Given a reference motion, we seek to design control strategies that track the reference motion while responding realistically to both unexpected perturbations and anticipated changes in the environment. Our approach to controlling an articulated character leverages the advantages offered by the modal coordinates. We apply modal analysis to the linearized N-DOF dynamic system (Equation 84), resulting in N decoupled one-dimensional equations. This transformation allows us to compute the control forces for each decoupled mode independently. We now classify these modes into three categories and develop separate control strategies for each type (Section 4.5).

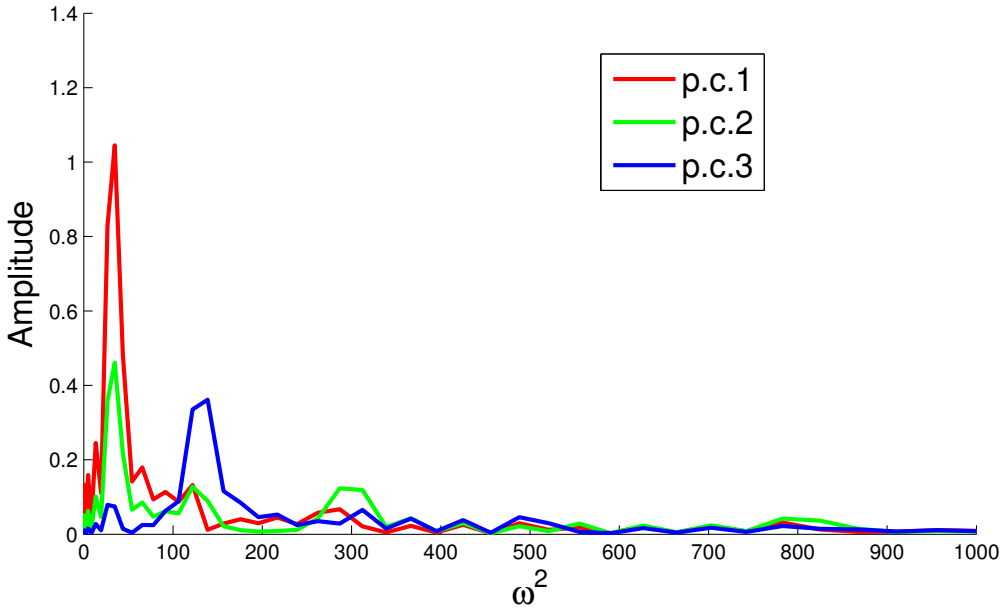
1. **Rigid body modes**: Because the global DOFs are not equipped with springs, the stiffness matrix K in Equation 84 is always singular. These under-actuated

DOFs result in six eigenvectors in Φ that correspond to zero eigenvalues. These six eigenvectors, called the rigid body modes, are only affected by external forces.

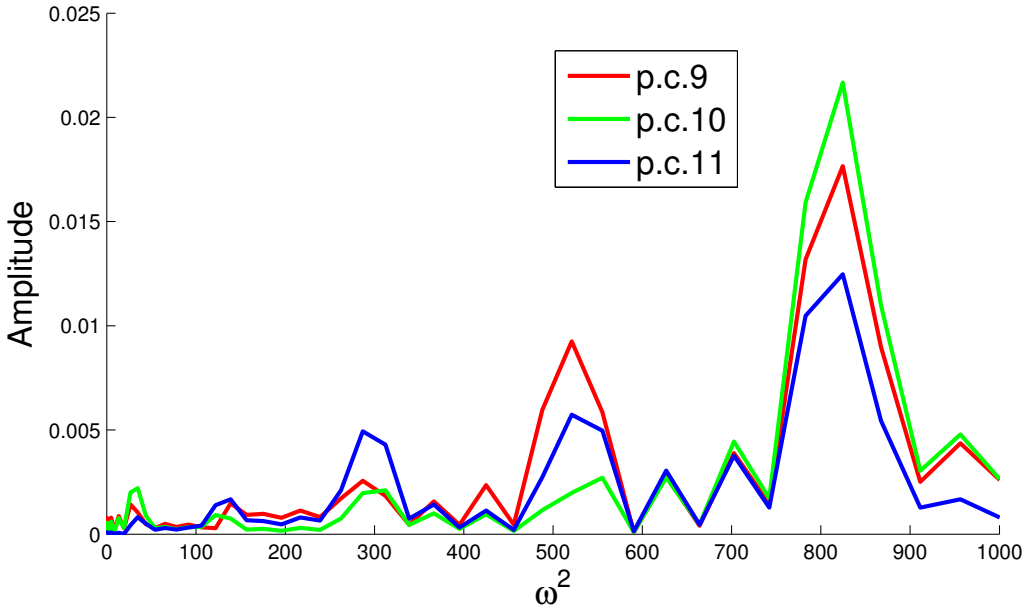
2. **Low frequency modes**: Except for the rigid body modes, all the remaining modes of the articulated system are actuated. For those modes with corresponding eigenvalue less than a chosen threshold, we classify them as low frequency modes. The choice of this threshold depends on the application and is discussed in Section 4.7. The motivation to focus on low frequency modes is due to the observation that visually more significant movements in human motion typically correspond to low-frequency motion. In our experiments, we applied frequency analysis on recorded human motion projected on its principal components. The results show that the first few principal components, that capture most of the variations in the motion [75], have dominant lower frequencies (Figure 14a) while the last few have dispersed frequency components biased toward the high frequency range (Figure 14b). This implies that controlling low-frequency modes can effectively change visually significant movements of the character.

3. **High frequency modes**: All the remaining modes above the threshold belong to the high frequency modes. Comparing to the low frequency modes, the motion in the high frequency modes has a little visual effect to the appearance of the motion (Figure 14b). Because of the high stiffness, controlling the motion in these modes usually requires large forces.

Once we classify the modes for the linearized dynamic system around the current state of the character, we apply long-horizon planning to the rigid body and the low frequency modes and short-horizon planning to the high frequency modes. The choice of horizon for different modes is due to the following two reasons. First, because the long-term planning is more computationally costly, we only apply it on the modes



(a) Three most important principal components



(b) Three least important principal components

Figure 14: We apply principal component analysis on the poses of a captured walking sequence. The first three principal components have dominant low frequencies. The last three principal components have dispersed frequency content biased towards high frequency range. Note that the scales of the vertical axes in the plots are different for better illustration.

that can make significant visual differences, namely, rigid body and low frequency modes. Second, because the long-term planning allows for temporary deviation from the reference trajectory, applying it to high frequency modes can cause large corrective forces, leading to instability and unnatural oscillations.

4.5 Control formulation

We now discuss our formulation of control strategies for each class of modes. Our goal is to compute the required contact forces and joint actuation within a window of time, such that the character can reach the corresponding reference state at the end of the window. We choose a window size of $n = 1$ for the short-horizon planning and a larger number for the long-horizon planning (Section 4.7).

Notation. Our simulation discretizes the time domain with a fixed time step Δt. t_0 indicates the current time while $t_k = t_0 + k\Delta t$ is the time at k time steps in the future, where $k \in \mathbb{Z}^+$. The horizon for each planning problem is determined by the window size of n frames or $n\Delta t$ seconds. We represent the state of the character at time t_k as $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$. Similarly, the corresponding state of the reference motion is represented as $(\bar{\mathbf{q}}_k, \dot{\bar{\mathbf{q}}}_k)$.

Applying the modal transformation to Equation 84, we express the equation in the modal coordinates $\mathbf{p}$, with the linear invertible relation $\Delta\mathbf{q} = \Phi\mathbf{p}$. The modal transformation matrix Φ is updated at each time step based on the current state of the character, $\mathbf{q}_0$: $M^{-1}(\mathbf{q}_0)K = \Phi\Omega\Phi^{-1}$. The desired state at the end of the current window, $(\bar{\mathbf{q}}_n, \dot{\bar{\mathbf{q}}}_n)$, can be transformed into modal coordinates as $(\bar{\mathbf{p}}_n, \dot{\bar{\mathbf{p}}}_n) = (\Phi^{-1}(\bar{\mathbf{q}}_n - \mathbf{q}_0), \Phi^{-1}\dot{\bar{\mathbf{q}}}_n)$.

4.5.1 Estimate contact forces: rigid body modes

Because the rigid body modes are not equipped with actuators, they are directly controlled by the contact forces. Our goal is to compute the contact forces such that,

at the end of the planning horizon t_n, the states of rigid body modes are as close as possible to the desired states. To express the contact forces in the modal coordinates, first we must determine the number of contact points at each time step within the planning window $[t_0, t_n]$. The contacts at the current frame t_0 is determined by a collision detection routine. For any other time step in the window, t_k, $k = 1, \ldots, n$, the contact information is directly taken from the reference motion.

We denote the contact forces $\mathbf{f}_{k,i}, i = 1 \ldots n_k$ at a given time t_k, where n_k denotes the number of contact points at time t_k. The sum of these contact forces in the modal coordinates can be expressed as $\Phi^T \sum_i J_i^T \mathbf{f}_{k,i}$, where $J_i = \frac{\partial \mathbf{x}_i}{\partial \mathbf{q}_0}$. J_i is the Jacobian evaluated at the point of application $\mathbf{x}_i$. If we assume that the contact forces hold fixed over a small interval of time Δt, we can approximate the effect of the contact forces by an impulse $\mathbf{I}_k^c = Z_k \mathbf{f}_k$, where $Z_k = \Delta t \Phi^T \mathbf{J}_k^T$, $\mathbf{J}_k = [J_1^T, \ldots, J_{n_k}^T]^T$ and $\mathbf{f}_k = (\mathbf{f}_{k,1}^T, \ldots, \mathbf{f}_{k,n_k}^T)^T$.

Based on Equation 81 and Equation 82 , the states of rigid body modes at time t_n under the influence of contact forces can be expressed as:

$$\mathbf{p}_n^r = \tilde{\mathbf{p}}_n^r + \sum_{k=0}^{n-1} A^r(t_n - t_k)\mathbf{I}_k^c \tag{85}$$

$$\dot{\mathbf{p}}_n^r = \dot{\tilde{\mathbf{p}}}_n^r + \sum_{k=0}^{n-1} \dot{A}^r(t_n - t_k)\mathbf{I}_k^c \tag{86}$$

These modal state equations are linear in $\mathbf{f}_k$'s. Now, we formulate an optimization problem to solve for the optimal contact forces $\mathbf{f}^*$ and minimize the state deviation and the contact force magnitude:

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} \|\mathbf{p}_n^r - \bar{\mathbf{p}}_n^r\|_{W_1}^2 + \|\dot{\mathbf{p}}_n^r - \dot{\bar{\mathbf{p}}}_n^r\|_{W_2}^2 + \|\mathbf{f}\|_{W_3}^2 \tag{87}$$

where $\mathbf{f}$ represents all the contact forces $(\mathbf{f}_0^T, \ldots, \mathbf{f}_{n-1}^T)^T$ and W_1, W_2 and W_3 are positive diagonal weighting matrices (see details in Section 4.7). $\|\mathbf{v}\|_W$ denotes the norm $(\mathbf{v}^T W \mathbf{v})^{\frac{1}{2}}$ for any vector $\mathbf{v}$. Equation 87 is an unconstrained convex quadratic programming (QP) problem and can be solved efficiently. This formulation is valid for bilateral contacts such as a hand grasp.

Coulomb friction. We apply unilateral forces for the ground contacts. The contact forces are constrained within a cone defined by friction coefficient μ. Assuming static contact, we define a contact force $\mathbf{f}$ by its component f along the contact normal $\hat{\mathbf{n}}$ and tangential component $\mathbf{f}_T$ such that $\|\mathbf{f}_T\| \leq \mu f$. We approximate $\mathbf{f}_T$ by a set of unit vectors $\hat{\mathbf{d}}_j$s.These unit vectors span the tangential plane as described in [84, 13]. We arrange these vectors as columns of matrix D. The coefficients corresponding to the unit vectors $\hat{\mathbf{d}}$'s are represented by $\boldsymbol{\beta}$ (with $\boldsymbol{\beta} \geq \mathbf{0}$). The legal contact force can be approximated by a polyhedral friction cone:

$$\mathbf{f} = f\hat{\mathbf{n}} + D\boldsymbol{\beta} \quad \text{with} \sum_j \beta_j \leq \mu f \quad \text{and} \quad f, \boldsymbol{\beta} \geq \mathbf{0} \tag{88}$$

Now, substituting Equation 88 in Equation 87, and adding the boundary conditions, we get a constrained convex QP problem in $f, \boldsymbol{\beta}$:

$$\begin{aligned} \min_{f,\boldsymbol{\beta}} \quad & \|\mathbf{p}_n^r - \bar{\mathbf{p}}_n^r\|_{W_1}^2 + \|\dot{\mathbf{p}}_n^r - \dot{\bar{\mathbf{p}}}_n^r\|_{W_2}^2 + \|\mathbf{f}\|_{W_3}^2 \\ \text{subject to} \quad & \mu f - \mathbf{e}^T\boldsymbol{\beta} \geq 0 \quad \text{with } f, \boldsymbol{\beta} \geq \mathbf{0} \end{aligned} \tag{89}$$

where $\mathbf{e} = [1, \ldots, 1]^T$.

4.5.2 Estimate joint actuation: low frequency modes

The low frequency modes can be controlled by the contact forces, as well as their own actuators. Given the contact forces $\mathbf{f}^*$ optimized for the rigid body modes, the control strategy for the low frequency modes is to optimize the actuation such that, at the end of the planning horizon t_n, the states of the low frequency modes match the desired reference states.

The state of these modes at time t_n under the influence of actuation impulses $\mathbf{I}^a$ and optimized contact forces $\mathbf{f}^*$ is given by:

$$\mathbf{p}_n^l = \tilde{\mathbf{p}}_n^l + \sum_{k=0}^{n-1} A^l(t_n - t_k) Z_k \mathbf{f}_k^* + \sum_{k=0}^{n-1} A^l(t_n - t_k)\mathbf{I}_k^a \tag{90}$$

$$\dot{\mathbf{p}}_n^l = \dot{\tilde{\mathbf{p}}}_n^l + \sum_{k=0}^{n-1} \dot{A}^l(t_n - t_k) Z_k \mathbf{f}_k^* + \sum_{k=0}^{n-1} \dot{A}^l(t_n - t_k)\mathbf{I}_k^a \tag{91}$$

Again, these equations are linear in $\mathbf{I}^a$. In addition, because matrix $A^l(t)$ is diagonal, these equations can also be decoupled. We rewrite the equations for m^{th} low frequency mode as:

$$p^l_{n,m} = c^l_{n,m} + \sum_{k=0}^{n-1} a^l_m (t_n - t_k) I^a_{k,m} \tag{92}$$

$$\dot{p}^l_{n,m} = \dot{c}^l_{n,m} + \sum_{k=0}^{n-1} \dot{a}^l_m (t_n - t_k) I^a_{k,m} \tag{93}$$

where a^l_m and $\dot{a}^l_m$ are the diagonal elements of A^l and $\dot{A}^l$ respectively, corresponding to mode m. $\mathbf{c}^l_n$ and $\dot{\mathbf{c}}^l_n$ indicate the first two terms of Equation 90 and Equation 91 respectively.

Now, we formulate an optimization to solve for actuation $\mathbf{I}^a_m = (I^a_{0,m}, \ldots, I^a_{n-1,m})^T$ for each mode m independently. Our goal is use the least amount of actuation to track a future state of the reference motion:

$$\mathbf{I}^{a*}_m = \underset{\mathbf{I}^a_m}{\operatorname{argmin}}\, w_4 (p^l_{n,m} - \bar{p}^l_{n,m})^2 + w_5 (\dot{p}^l_{n,m} - \dot{\bar{p}}^l_{n,m})^2 + \|\mathbf{I}^a_m\|^2_{W_6} \tag{94}$$

where w_4 and w_5 are scalar weights for position and velocity matching respectively, and W_6 is a positive diagonal matrix. This is again an unconstrained convex QP problem. Since these problems are uncoupled for all the modes, we can solve them independently and efficiently.

4.5.3 Track high frequency modes

We use a short-horizon to plan the control forces for the high frequency modes, namely, $n = 1$. Similar to Equation 92, the state of each high frequency mode at the next state can be expressed as $p^h_{1,m} = c^h_{1,m} + a^h_m (t_1 - t_0) I^a_{0,m}$. Since there is only one control variable, $I^a_{0,m}$, we can solve it analytically:

$$I^{a*}_{0,m} = \frac{\bar{p}^h_{1,m} - c^h_{1,m}}{a^h_m (\Delta t)} \tag{95}$$

4.5.4 Enforce physical contact model

If we directly apply the optimal contact forces $\mathbf{f}_0^*$ and actuation impulses $\mathbf{I}_0^{a*}$ at the current time step, we can analytically compute the modal state at the next time step t_1 by the virtue of Equation 81:

$$\mathbf{p}_1^* = \tilde{\mathbf{p}}_1 + A(\Delta t)(Z_0\mathbf{f}_0^* + \mathbf{I}_0^{a*}) \tag{96}$$

However, the estimated state $\mathbf{p}_1^*$ might cause artifacts at contact points, such as slipping, penetration, and breakage of contacts. We formulate a linear complementarity problem (LCP) to solve this issue. Specifically, we adjust the contact forces by $\Delta\mathbf{f} = (\Delta\mathbf{f}_1^T, \ldots, \Delta\mathbf{f}_{n_0}^T)^T$ and the joint actuation by $\Delta\mathbf{I}^a$, such that the movement of each contact point, $\Delta\mathbf{x}_i \approx J_i\Delta\mathbf{q} = J_i\Phi\mathbf{p}_1$, and the corrective forces, $\Delta\mathbf{f}$ and $\Delta\mathbf{I}^a$, satisfy the Coulomb friction model. The modal state at the next time step, $\mathbf{p}_1$, is then computed as:

$$\mathbf{p}_1 = \mathbf{p}_1^* + A(\Delta t)(Z_0\Delta\mathbf{f} + \Delta\mathbf{I}^a) \tag{97}$$

Control preference. Because there can be many solutions for the corrective contact forces $\Delta\mathbf{f}$ and the corrective joint actuation $\Delta\mathbf{I}^a$ that satisfy the Coulomb friction model, we can formulate an optimization with an objective function, $E(\Delta\mathbf{f}, \Delta\mathbf{I}^a)$, that minimizes a certain desired criterion.

However, we cannot directly include this objective function in the LCP formulation (LCP is not an optimization problem). If we choose a quadratic form for E, the LCP solution can be biased towards the minimum of E, by deriving a linear relation between $\Delta\mathbf{I}^a$ and $\Delta\mathbf{f}$ using the optimality condition. We can then express $\Delta\mathbf{I}^a$ in terms of $\Delta\mathbf{f}$ in the LCP formulation. In our implementation, we choose to minimize the impact of corrective forces in the state of rigid body modes and the low frequency modes:

$$E(\Delta\mathbf{f}, \Delta\mathbf{I}^a) = \|A^r(\Delta t)Z_0\Delta\mathbf{f}\|_{W_r}^2 + \|A^l(\Delta t)(Z_0\Delta\mathbf{f} + \Delta\mathbf{I}^a)\|_{W_l}^2 \tag{98}$$

Based on the optimality condition, the gradients of E vanish at the minimum. We obtain a linear relation between $\Delta\mathbf{I}^a$ and $\Delta\mathbf{f}$:

$$P\Delta\mathbf{I}^a + Q\Delta\mathbf{f} = 0$$

The coefficient matrix of $\Delta\mathbf{I}^a$ may not be full rank and invertible, but we can solve this issue by reformulating the objective function:

$$\min_{\Delta\mathbf{I}^a} \|P\Delta\mathbf{I}^a + Q\Delta\mathbf{f}\|^2 + \|\Delta\mathbf{I}^a\|^2_{W_a} \tag{99}$$

where W_a is a positive definite weighting matrix. We now solve for the corrective actuation $\Delta\mathbf{I}^a$ that minimizes the error in optimality condition of E and the magnitude of $\Delta\mathbf{I}^a$. The optimality condition of the new objective function is given by a linear relation:

$$\begin{aligned} \Delta\mathbf{I}^a &= X_f\Delta\mathbf{f} \\ \text{where } X_f &= -(P^TP + W_a)^{-1}P^TQ \end{aligned} \tag{100}$$

Note that X_f is well defined since $P^TP + W_a$ is always positive definite. This is a similar treatment as in [69], in that the relation was derived based on matching the acceleration of certain chosen features.

For the high frequency modes, we would like to maintain the planned actuation computed in Section 4.5.3. We define the components corresponding to the high frequency modes in X_f such that for m^{th} high frequency mode, $\Delta I^a_m = -(Z_0\Delta\mathbf{f})_m$ resulting in $p_{1,m} = p^*_{1,m}$.

We can now rewrite the modal state $\mathbf{p}_1$ in Equation 97 as:

$$\mathbf{p}_1 = \mathbf{p}^*_1 + A(\Delta t)(Z_0 + X_f)\Delta\mathbf{f} \tag{101}$$

Note the only explicit variables are the corrective contact forces, $\Delta\mathbf{f}$. We now can formulate a LCP based on the following constraints derived from Coulomb friction model.

Unilateral contact force. Breaking down each force in normal and tangential components (as in Equation 88), we have:

$$\mathbf{f}^*_{0,i} + \Delta\mathbf{f}_i = f_i\hat{\mathbf{n}}_i + D_i\boldsymbol{\beta}_i, \quad \text{with} \quad f_i, \boldsymbol{\beta}_i \geq \mathbf{0} \tag{102}$$

Normal constraints. In the normal direction, the movement of the contact point $\mathbf{x}_i$ is given by $\hat{\mathbf{n}}_i^T\Delta\mathbf{x}_i$. The following complementarity conditions must be satisfied for each contact:

$$(\hat{\mathbf{n}}_i^T\Delta\mathbf{x}_i)f_i = 0 \text{ with } f_i \geq 0, \hat{\mathbf{n}}_i^T\Delta\mathbf{x}_i \geq 0 \tag{103}$$

Tangential constraints. In the tangential direction, we need to ensure that if the point moves, the contact force lies on the boundary of the cone and in the direction opposite to the movement. Using the formulation similar to [84, 13], we introduce a parameter λ_i that represents the relative tangential movement of the contact point. Therefore, the constraints are written as:

$$(\lambda_i\mathbf{e} + D_i^T\Delta\mathbf{x}_i)^T\boldsymbol{\beta}_i = 0 \quad \text{with} \quad \lambda_i\mathbf{e} + D_i^T\Delta\mathbf{x}_i \geq \mathbf{0}, \boldsymbol{\beta}_i \geq \mathbf{0} \tag{104}$$

$$(\mu f_i - \mathbf{e}^T\boldsymbol{\beta}_i)\lambda_i = 0 \quad \text{with} \quad \mu f_i - \mathbf{e}^T\boldsymbol{\beta}_i \geq 0, \lambda_i \geq 0 \tag{105}$$

Colloquially, Equation 104 enforces that the direction of movement is opposite to the friction force and Equation 105 enforces that the friction force is on the boundary if the point of contact moves.

Linear complementarity problem (LCP). We define $\mathbf{z} = (\mathbf{z}_1^T, \ldots, \mathbf{z}_{n_0}^T)^T$ where $\mathbf{z}_i = (f_i, \boldsymbol{\beta}_i^T, \lambda_i)^T$ and rewrite Equation 102 as $\Delta\mathbf{f}_i = N_i\mathbf{z}_i - \mathbf{f}^*_{0,i}$, where $N_i = [\hat{\mathbf{n}}_i, D_i, 0]$. Stacking all the points together, we write:

$$\Delta\mathbf{f} = N\mathbf{z} - \mathbf{f}_0^* \tag{106}$$

where N is a block diagonal consisting of N_i's. Using Equation 101 and Equation 106, we can write $\mathbf{p}_1$ as a function of $\mathbf{z}$:

$$\mathbf{p}_1 = (\mathbf{p}_1^* - A(\Delta t)(Z_0 + X_f)\mathbf{f}_0^*) + (A(\Delta t)(Z_0 + X_f)N)\,\mathbf{z} \tag{107}$$

Recalling $\Delta\mathbf{x}_i \approx J_i\Delta\mathbf{q} = J_i\Phi\mathbf{p}_1$ and using Equation 103, Equation 104 and Equation 105, we write the complementarity conditions for each contact point $\mathbf{x}_i$ as $\mathbf{w}_i = F_i\mathbf{p}_1 + G_i\mathbf{z}_i$ and $\mathbf{w}_i^T\mathbf{z}_i = 0$ with $\mathbf{w}_i, \mathbf{z}_i \geq \mathbf{0}$, where

$$F_i = \begin{pmatrix} \hat{\mathbf{n}}_i J_i\Phi \\ D_i^T J_i\Phi \\ 0 \end{pmatrix} \text{ and } G_i = \begin{pmatrix} 0 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{e} \\ \mu & -\mathbf{e}^T & 0 \end{pmatrix}$$

Stacking the equations for all the contact points together, we get $\mathbf{w} = F\mathbf{p}_1 + G\mathbf{z}$ and $\mathbf{w}^T\mathbf{z} = 0$ with $\mathbf{w}, \mathbf{z} \geq \mathbf{0}$, where matrices $F = (F_1^T, \ldots, F_{n_0}^T)^T$ and $G = \text{blockdiag}(G_1, \ldots, G_{n_0})$. Substituting Equation 107 in the above we get the following LCP problem:

$$\mathbf{w} = C\mathbf{z} + \mathbf{h} \text{ and } \mathbf{w}^T\mathbf{z} = 0 \text{ with } \mathbf{w}, \mathbf{z} \geq \mathbf{0} \tag{108}$$

where

$$\begin{aligned} C &= FA(\Delta t)(Z_0 + X_f)N + G \\ \mathbf{h} &= F\left(\mathbf{p}_1^* - A(\Delta t)(Z_0 + X_f)\mathbf{f}_0^*\right) \end{aligned}$$

Once the solution $\mathbf{z}$ is obtained, we compute the required forces using Equation 106 and Equation 100. Finally, we compute the modal position at next time step $\mathbf{p}_1$ using Equation 97 and the next pose $\mathbf{q}_1$ as $\Phi\mathbf{p}_1$.

4.5.5 Summary

We summarize the control procedure that advances the character from the current time step to the next. Starting out with a new current state t_0, we solve the control forces for a time window $[t_0, t_n]$ by following steps:

- Solve for the ideal contact forces $\mathbf{f}^*$ such that the components of the state of the rigid body modes at t_n match the corresponding reference state.

- Given $\mathbf{f}^*$, solve for the ideal actuation for the low frequency modes $\mathbf{I}^{a*}$ such that the state of the low frequency components at t_n matches the corresponding reference state.

- Compute actuation for the high frequency modes such that the state of the high frequency modes at t_1 exactly matches the next reference state.

- Compute the corrective forces $\Delta\mathbf{f}$ and $\Delta\mathbf{I}^a$ to satisfy the Coulomb friction model.

Once the corrective forces are added to the ideal forces, we get the state at the next time step $\mathbf{q}_1$. We then advance the time step and repeat the same procedure. The ideal forces computed for the rest of the window have to be recomputed again at the next time step for two reasons. First, the environment state can change at any moment and the character must adapt her plan accordingly. Second, the ideal forces are computed based on a locally linearized dynamic system. Replanning ensures that more accurate control forces are applied to the character.

The control and simulation are performed with respect to linearized dynamics and no numerical integration is needed since the next state can be computed analytically using Equation 97.

4.6 Interaction and motion editing

To create variations from the original reference motion, our system allows the user to apply external forces to the character, as well as directly modify the reference trajectory on the fly.

Perturbations. We assume that the character reacts to unexpected perturbations with 200 ms latency to simulate the muscle activation delay in the sensory feedback [66, 32]. We denote the response time as t_r and perturbation time as t_0. For a simulation time t_k in the time interval $[t_0, t_0 + t_r]$, we replace the current state of the character, $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$, by the corresponding state in reference motion, $(\bar{\mathbf{q}}_k, \dot{\bar{\mathbf{q}}}_k)$ for the

purpose of computing control forces in the rigid body and the low frequency modes. During this response interval, the control system makes a long-term plan as if the character has not sensed the external force. Once the control forces for the rigid body and the low frequency modes are computed, we restore the current state $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$ and compute the forces for the high frequency modes and the corrective forces for the contacts. After the response interval, we compute the control forces in the usual manner to recover back to the reference motion.

Motion editing. When the character responds to larger perturbations or changes behaviors volitionally, tracking the original reference trajectory becomes a poor control strategy. Since our control algorithm does not require any offline computation based on the reference trajectory, we are free to edit the reference motion on the fly. In theory, any online trajectory editing technique can be applied, we implemented following three generic methods:

- Forward and inverse kinematics (FK/IK): The user can directly change the joint angles of the reference motion via FK or modify the position of a body point via IK.
- Time warping: The user can select parts of the motion and change their speed.
- Motion transition: The user can select a new motion sequence and blend the current state of the character into the new motion over a time interval.

4.7 Results

We now present our results and report all the parameters for controlling a 3D human articulated character under perturbations and changes of the environment. Full animations can be seen in the supplemental video available online in the ACM Digital Library [40].

Character description. The skeleton for our articulated character consists of 18 rigid links and 42 DOFs. We define the stiffness for each DOF as the total mass of the subtree rooted at the joint where the DOF resides. We use zero damping in all our examples. The threshold for classifying the modes (Section 4.4) is chosen as 100 $(rad/s)^2$. For our articulated character, this choice of threshold results in 14 low frequency modes and 22 high frequency modes apart from 6 rigid body modes. i.e. we use around half of the total number of modes to formulate the long term planning problem and track the motion for the rest of the modes using a short term plan.

Global constants. We use time step $\Delta t = 1.0/120\ s$ for all our examples. The friction cone in Equation 88 is approximated using six basis vectors defining a hexagonal boundary for the cone. The friction coefficient μ is chosen to be 1.0 for all the simulations.

Control parameters. We use the same set of weights in the control algorithm for synthesizing all the motions sequences. For Equation 89, we define W_1 as a diagonal matrix with first three components corresponding to global positions as 0.1 and the remaining three corresponding to global rotations as 0.5. The matrix W_2 is an identity matrix, I, and $W_3 = 0.1I$. For Equation 94, we choose $w_4 = 0.5$, $w_5 = 2.0$ and $W_6 = 5 \times 10^{-3} I$. In our experience, penalizing velocity terms more than the positions gives more robust solution for control. Finally, we choose the weighting matrix in Equation 99 as $W_a = 10^{-3} I$.

We use MOSEK (`mosek.com`) to solve the convex QPs formulated in Equation 89 and Equation 94. We solve the LCP in Equation 108 through a C++ interface to MATLAB's '*lcprog*' solver. All the results are synthesized on a single core of Intel Core 2 Duo 2.8 GHz processor.

4.7.1 Tracking a reference motion

With all the global constants and control parameters defined, the only parameter specific to each sequence is the window size for long-term planning.

We choose a window size of 12 frames to synthesize a biped character walking on a flat surface while tracking a mocap reference motion. In our experience, window sizes ranging from $10-15$ work well with walking motion. We synthesize walking motion at $4-10$ frames per second (fps) or 3-10% real-time speed. The bottle neck in control computation is solving the QP for the rigid modes (Equation 89) due to a large number of contact force parameters, and takes more than 80% of the computation time at every time step. The rest of the computation involves eigenvalue decomposition for modal analysis, QP solution for low frequency modes actuation (Equation 94) and LCP for contact correction.

We use a window size of 16 frames for tracking a squatting motion and 24 frames for a swinging motion. For these motions, we simplified the QP problem for the rigid modes by choosing a single force for a contact point that lasts for a few frames in the window, thereby largely reducing the number of unknowns. As a result, the control algorithm can reach $30-40$ fps or 25-30% real-time speed.

4.7.2 Responding to external perturbations

To demonstrate that the character is able to passively respond to arbitrary external perturbations, we synthesize responses of the character to user-applied forces while performing different tasks, such as walking, squatting, and swinging. When pushed at different body parts with different direction and magnitude of forces, the character reacts passively and eventually recovers (Figure 15). We apply forces varying between $100-175$ N on walking motions for a period of 10 frames, $75-250$ N on squatting for 20 frames and $75-175$ N on swinging for 20 frames.

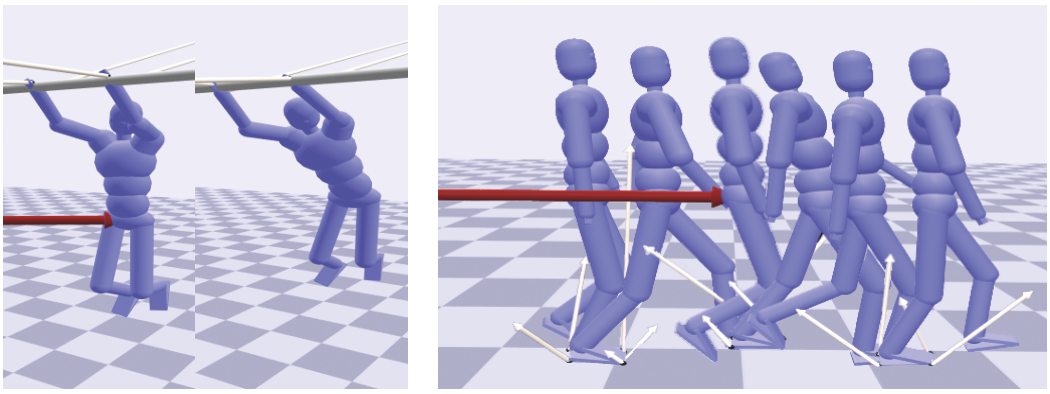


Figure 15: Perturbation responses while performing different tasks.

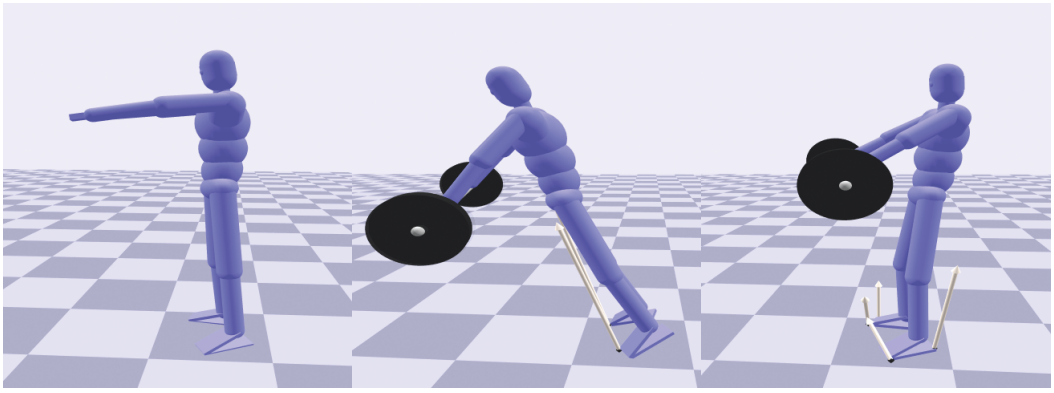
4.7.3 Editing the reference motion

Our control algorithm also allows the reference motion to be modified online. We edit the reference motion using following three editing techniques.

FK/IK. We use a very simple online editing technique to modify the original walking motion to walking on a ramp. The editing only involves changing the vertical root translation to align with the slope and using IK to match the feet to the surface of the ramp. We synthesize motions walking up the slope of up to 3 degrees and walking down the slope up to 5 degrees. Similarly, we can modify the length of the strides in walking motion by changing the horizontal root translation and using IK to maintain the original duration of contacts. With these simple edits, we can increase the stride lengths by 10 cm and reduce it by 15 cm for every step of the reference motion.

To demonstrate more drastic changes on the reference motion, we modify the squatting motion to a weight-lifting motion by applying IK to the hands so they come into contact with a 30 kg barbell. The edited motion looks very unrealistic because the modification does not consider the changes in dynamics. However, the motion simulated by our system appears more natural as the character struggles to balance when she lifts the bell. In addition, we compare the results with two long-term window sizes of 12 and 36 frames (0.1 s and 0.3 s resp., Figure 16). Simulating with

12 frame window, the character picks up the barbell but topples forward eventually. Increasing the window size to 36 frames allows the character to plan control forces further ahead of time. As a result, the character chooses to lean backward to balance the weight and remains steady after picking up the weight.

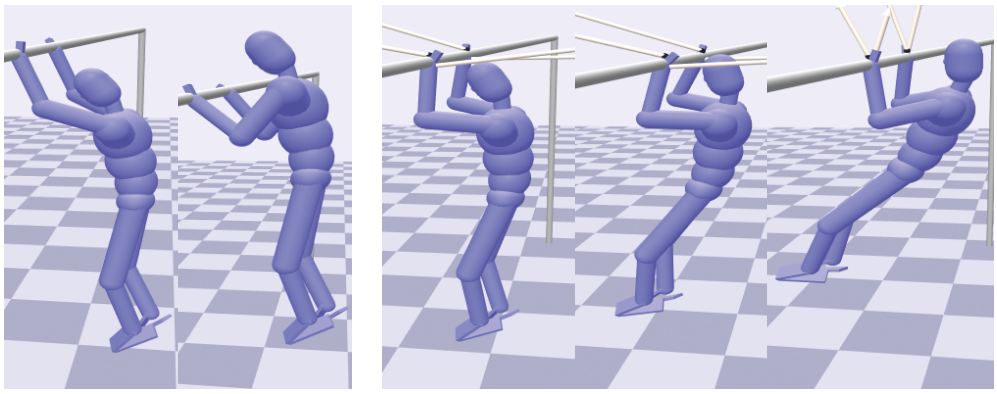


(a) original (b) shorter window (c) longer window

Figure 16: Weight-lifting simulation with different window sizes.

In addition to editing existing mocap sequences, we synthesize a completely new motion by interpolating a few edited keyframes. Starting out from a single pose of character hanging from the bar, we create a very rough chin-up motion by changing the vertical root position and applying IK to maintain the hand positions (Figure 17a). The character in this hand-crafted reference motion appears stiff and unnaturally strong. In contrast, the output motion exhibits realistic dynamics and responses to the external perturbations (Figure 17b).

Time warping. In addition to spatially editing the motion trajectories, we also edit the timing to speed up or slow down parts of a motion as desired. To demonstrate this, we speedup the walking motion by 10% and simultaneously increase the stride length by 15 *cm* per step. The character is able to walk steadily while tracking this edited motion. Time warping can also be used to aid in better recovery from unexpected perturbations. We give a strong backward push to a walking character. Losing forward momentum, the character fails to walk soon after the push because



(a) key frames (b) simulation

Figure 17: Simulation of a chin-up exercise from two key frames.

she is not able to keep up with the timing of the original motion. We repeat the experiment with time warping immediately after the push to let the character recover back to original motion. We slow down the reference motion by 25% for an interval of 1 second after the push. Now, the character is able to recover and walk steadily with a different phase as a result of warping.

Motion transition. Finally, we demonstrate tracking two different reference motions in sequence. We edit a jumping motion of the character by raising her arms. When the character is airborne, we add constraints on the hands to simulate bilateral contacts with a high-bar. The character passively swings on the bar while still tracking the jumping motion. We then transition into a swinging motion by blending into a swing reference motion over a time interval of 0.5 s. Just before the transition, the pose of the character differs from the reference motion including the contact points on the hand. We seamlessly synthesize the transition and the character starts tracking the swinging motion.

4.8 Discussion

To demonstrate the effectiveness of our modal analysis approach, we conducted a few quantitative experiments and produced qualitative results to support the following

two claims. First, we argue that dimension reduction is critical for online (re)planning and modal analysis approach is very efficient due to modal decoupling. Second, we argue that using natural frequency to select controlled modes is appropriate for human motion. We also address a few limitations of our work in this section.

4.8.1 Analysis

Optimal control for a long horizon is computationally expensive and typically prohibitive for online applications. A standard method for long-term optimal control in computer animation is to formulate a spacetime optimization problem. For our problem, a 12 frame window of spacetime results in more than 700 unknowns and 70 constraints. This highly nonconvex optimization requires computation of several Jacobians and Hessians for every frame, leading to an excruciatingly slow motion synthesis method at a rate of several minutes per frame. On the other hand, modal analysis reduces computational time by transforming the optimal control problem into a low dimensional space. Furthermore, modal analysis exploits the fact that the equations of motion can be decoupled, which significantly improves the performance in addition to the speed gain from dimension reduction. The details of performance can be found in Section 4.7.1.

For online long-term optimal control, the necessity of dimension reduction is evident, but the selection of the low-dimensional space is also critical. Similar to PCA and other spectral embedding methods, modal analysis reduces the domain to a subspace spanned by a subset of modes, which can be organized in a specific order. For modal analysis, the modes are ordered based on the natural frequencies of the physical model. We conduct two experiments to show that mode selection based on natural frequency can effectively control human motion.

Our algorithm applies long-term control only in low frequency modes. To justify the choice of this control scheme, we compare the motions generated by selecting

different frequency ranges. In the first experiment, our algorithm selects the lowest frequency modes (mode 6 to 18) to simulate a walking sequence. This baseline is compared against motion sequences generated with control modes numbered from $6 + k$ to $18 + k$, where $k \in \mathbb{Z}^+$. The error metric used for comparison is defined as the norm of the difference in the global orientation in the simulated and the input motion. Figure 18 shows sequences controlled by the modes in different frequency ranges. We notice that the error increases and the system becomes more unstable as the value of k in increased. We also simulate a few motion sequences with randomly selected modes. The simulation becomes unstable quickly after a few frames. These results suggest that controlling the low frequency modes leads to the most stable control algorithm.

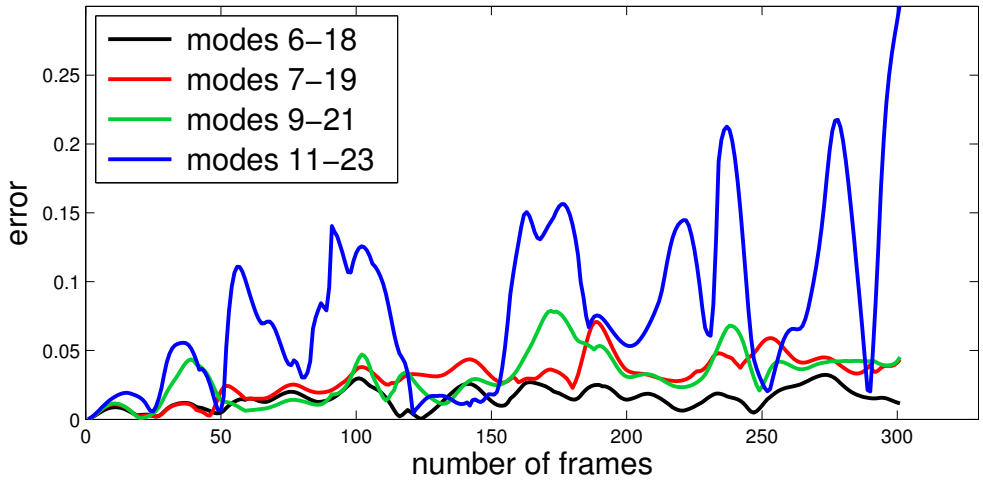


Figure 18: Error comparison for control of different set of modes

As we demonstrated, dimension reduction based on natural frequency is a viable approach for long-term, online optimal motion control, but what is the cutoff threshold that defines "low" frequency? In the second experiment, we analyze our choice of the threshold for classifying the modes as low frequency. Our chosen threshold works for all of the demonstrated examples. In our experience, the control algorithm works successfully within a variance of 3-4 modes from the chosen threshold. If the

chosen threshold is too small, most of the modes are categorized as the high frequency modes, resulting in very kinematic, less responsive motion. On the other hand, if the threshold is too large the resulting motion becomes very jerky and unstable. This is because the objective function in our control policy in Equation 94 penalizes the actuation in the modes below the threshold. However, modes with high frequency require large forces to execute the desired motion; hence our control policy for low frequency modes is clearly not suited for these modes.

The recent work of Ye and Liu [103] shares some common goals of our method of simulating a given reference motion and its online variations using a generic control method. They perform an offline control computation based on a given reference motion that restricts their method to handle small changes during online simulation. In contrast, our completely online control computation handles larger changes in the reference motion. To compare the performance of these two methods, we analyze two examples. First, walking motion with perturbations and second, squatting motion while lifting heavy weights. In the first example, it is not necessary to change the reference motion online since the perturbations are relatively small. Therefore, the computation time for method in [103] consists of one time offline computation of 1-2 minutes and online feedback computation that runs at an average of 20 fps. Our online control computation runs at 4-10 fps. For the second example of squatting while lifting weights, it is required for the method in [103] to change the reference trajectory to successfully perform the motion. The control has to be re-evaluated frequently that would make their computation prohibitively expensive during online simulation. In our method, the control computation runs at 30-40 fps thereby outperforming the other method.

4.8.2 Limitations

Despite the advantages offered by our approach, it suffers from a few limitations. Our control algorithm works better for physically correct reference motions. More challenging activities, such as walking, require a higher quality reference motion than other activities, such as squatting. To use motion capture data as reference, we must ensure two conditions. First, we compute the contact forces required to achieve the global trajectories (six underactuated degrees of freedom) from the reference motion. The computed contact forces need not satisfy the dynamic and friction constraints for the entire reference motion, but the longest interval that violates these constraints must be less than 10-15 frames (at 120 fps). Second, the contact points should remain in contact with the environment for their expected duration. For the first condition, we simply use a motion capture data that looks physically plausible without glaring artifacts. For the second condition, we preprocess the motion by applying inverse kinematics. To utilize keyframe animation or physically inconsistent mocap data (violating above two conditions), we could employ the spacetime optimization process as used in [69] as a preprocess to produce higher quality reference motion. Online editing of the reference motion should also satisfy these two conditions. Our current editing method modifies the root position in accordance with the changes in the contact positions. This simple kinematic editing might result in motions that occasionally violate the equations of motion, but as long as the physically inconsistent frames do not stretch longer than 10-15 frames, our controller can successfully handle the edited motion.

Although our system can synthesize large variations from the reference motion by editing the reference motion online, the modification is artificially done by user intervention, rather than caused by a physical response. Since the modification of the reference motion does not incorporate high-level strategies based on human postural responses to physical perturbations, the character can only handle relatively

small pushes. To recover from large perturbations using drastically different control strategies, it is inevitable to modify the reference trajectory or use an entirely different trajectory. Our method suffers from the inherent drawback of tracking controller in that deviations have to be confined in a small neighborhood of the reference trajectory. However, since our controller does not rely on any offline computation based on the reference trajectory, we can directly modify the trajectory online, resulting motion largely different from the original reference trajectory (e.g. from squatting to weight lifting).

The contact information and the timing is obtained from the reference motion. In the event of perturbation, we continue to use the same contact information for planning, but the actual contacts might be different from those in the reference motion e.g. earlier or later heel strike for double support. To increase the robustness in recovering from larger perturbations and environment changes, we would like to design dynamic policies to estimate the contact position and the timing.

We use a simple rigid body to model the foot with its four corners as the contact points. We suspect that the forces generated in our system have significant discrepancy from those measured by the force sensing platforms due to this simplification in modeling of foot geometry and contacts. However, though the individual forces computed at the contact points may not be realistic e.g. the forces at the corners face opposite directions, the aggregated force strictly satisfies the following two physical constraints. First, the center of pressure lies within the convex hull formed by the contact points on the foot. Second, the net ground reaction force produced by the foot obeys the Coulomb friction law. These constraints are enforced by solving a LCP (Equation 108), which corrects the difference in expected contact forces by the character and the actual physical forces.

Our control algorithm does not run in real-time. The bottleneck is the estimation of contact forces for controlling the rigid body modes, especially when the number of

contacts is large. One possible solution is to replan less frequently for control forces.

Size of the planning window plays an important role in the control policy. Despite the apparent advantages of having larger window sizes for farther look ahead, approximation of dynamics for the entire window introduces errors that limits us from choosing arbitrarily large windows. For example, choosing window size of 1 frame is similar to having a per-frame PD control. In this case, the character is extremely stiff and fails to track under-actuated tasks. Increasing the window size significantly improves the stability and ability to handle perturbations. However, the quality of control starts to degrade if the window size is too large as large errors accumulate due to the approximation of the dynamic equations (i.e. linearization of dynamics around the current frame). In the current implementation, the window size is chosen manually for every motion. We would like to explore ways of choosing the optimal window size automatically and adaptively based on the reference motion and the simulated state.

Approximation of the dynamics equation in Equation 84 works well for demonstrated motions such as walking. This linearization around zero velocity may introduce larger errors in dynamics for motions involving high joint velocities. We would like to explore higher order approximations to the dynamic equations that are more suited to high velocity motions.

We demonstrated our examples based on an under-damped system, specifically with zero damping. However, our control design is generic to include over-damped system as well. We choose an under-damped system because it better captures the natural dynamics that exploits the passive elements of a character.

Currently, we manually choose the constant stiffness parameters for the articulated character irrespective of the reference motion. Inspired by biomechanics studies on how stiffness of the passive elements varies by different activities [30], we would like to automatically design the stiffness parameters of the character based on the reference

motion. Our design goal is to solve an inverse problem of modal analysis such that the desired motion resembles the motion of the biomechanical system vibrating at its natural frequencies.

CHAPTER V

SOFT CONTACTS FOR ROBUST CONTROL

In this chapter, we investigate the impact of the deformable bodies on the control algorithms for physically simulated characters. We hypothesize that ignoring the effect of deformable bodies at the site of contact negatively affects the control algorithms, leading to less robust and unnatural character motions. To verify the hypothesis, we introduce a compact representation for an articulated character with deformable soft tissue and develop a practical system to simulate two-way coupling between rigid and deformable bodies in a robust and efficient manner. We then apply a few simple and widely used control algorithms, such as pose-space tracking control, Cartesian-space tracking control, and a biped controller (SIMBICON), to simulate a variety of behaviors for both full-body locomotion and hand manipulation. We conduct a series of experiments to compare our results with the motion generated by these algorithms on a character comprising only rigid bodies. The evaluation shows that the character with soft contact can withstand larger perturbations in a more noisy environment, as well as produce more realistic motion.

5.1 Introduction

One of the fundamental simplifications that researchers in physics-based human motion synthesis make, is that motion is the product of an articulated rigid body system with actuated joints representing bones and active skeletal muscles. On the surface this abstraction does capture the most fundamental aspects of the human musculoskeletal system. Utilizing this assumption, researchers have developed several control algorithms that can synthesize movement for various tasks like balance and walking. Although these controllers work well in their specific problem domain,

they still cannot achieve the same level of agility the human body displays.

In this chapter we revisit the fundamental assumption that an articulated rigid body system, by itself, captures the fundamental properties that enable human-like motion. We focus on one aspect of the motion that is not captured by this simplified model: the contact with the environment primarily occurs through the soft tissue. This factor comes into play in any situation where there is a collision between the character and the environment. Collisions between rigid bodies usually result in sporadic contact points and highly discontinuous pressure distribution. Although a character consist of only rigid bodies is ideal for efficiently simulating human movement, we postulate that the simplified rigid contact model inadvertently increases the difficulty in controller design and results in unrealistic motion.

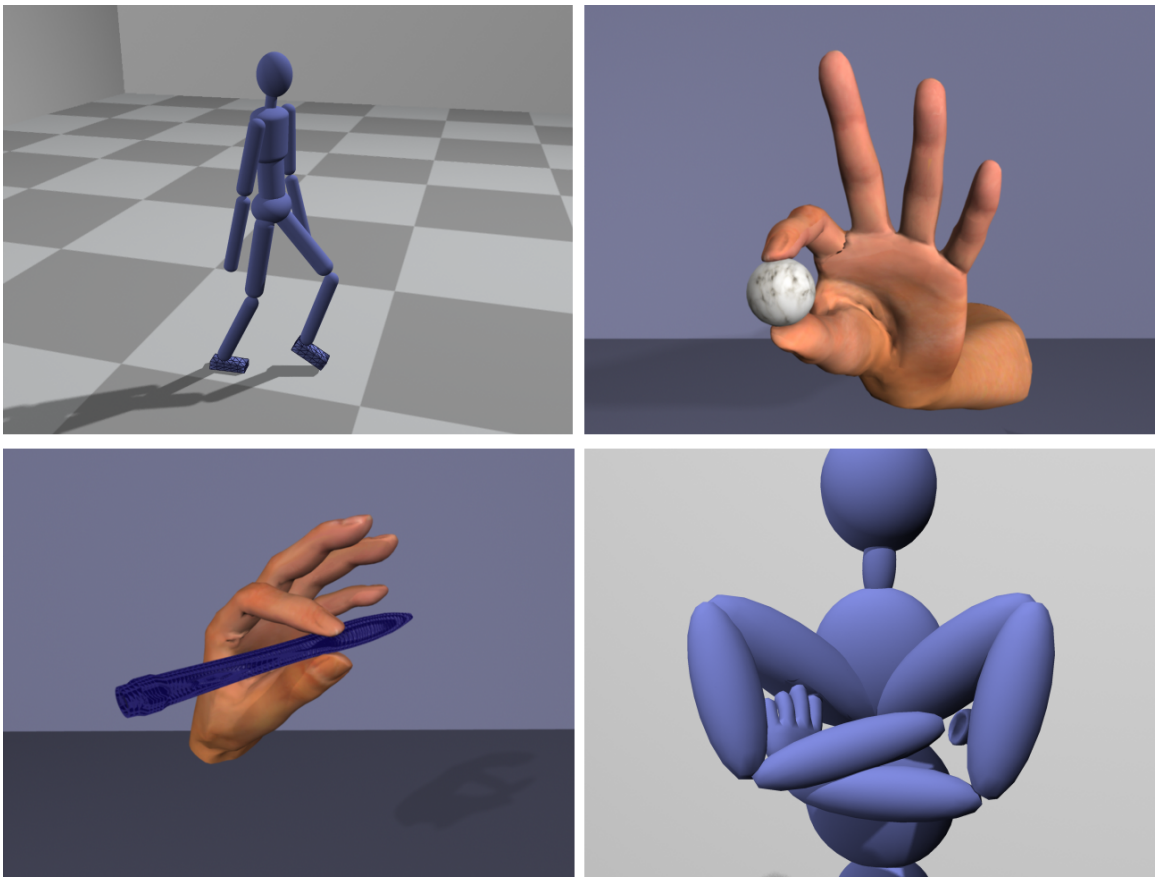


Figure 19: Various controllers for character animation can be improved by simulating soft tissue deformation at the site of contact.

The primary contribution in this chapter is to demonstrate that simple control strategies coupled with the simulation of soft tissue deformation at the site of contact can achieve very robust and realistic motion. We develop a practical system that allows us to simulate two-way coupling between rigid and deformable bodies in a robust and efficient manner. We then apply a few simple and widely used control algorithms, such as pose-space tracking control, Cartesian-space tracking control, and SIMBICON, to simulate a variety of both full-body locomotion and hand manipulation. The resulting motions are compared with the motion generated by these algorithms on a character comprising only of rigid bodies. These simple controllers demonstrate that the character with soft contacts can withstand larger perturbations in a more noisy environment, without the need of designing more sophisticated control algorithms.

Simulating deformable bodies can be achieved in a few different ways and the design choice often has to balance the required accuracy and performance. We hypothesize that the accuracy offered by sophisticated but expensive methods, such as Finite Element Method (FEM) is unnecessary for our application for two reasons. First, unlike most previous work that simulates deformation of complex volumetric meshes for aesthetic purpose, the primary goal of our work is to produce deformation for more physically correct contacts. Second, average human body deforms marginally due to the support of bones. In particular, the deformation due to contacts is typically small and localized. We take advantage of these properties to design a simple and accurate model that only computes the surface of deformable bodies, rather than the entire volume.

To this end, we introduce a new representation for human skeleton consist of an articulated rigid body system, in which each rigid body is surrounded by a set of point masses representing the surface of flesh. Each point mass is attached to the rigid body as a child link with three translational degrees of freedom (DOFs). The dynamics

of bones and flesh can be expressed in a unified manner by Lagrangian equations of motion in the generalized coordinates. This compact representation allows us to *soften* or *harden* any part of flesh at any time, by simply adding or removing the translational DOFs of involved point masses, without switching dynamic regimes or introducing any instability. Based on this flexible representation, we develop an efficient system where only the site of collision needs to be simulated as deformable body while the rest of the character remains rigid.

5.2 Related Work

Recent work in physics-based character animation explored a variety of approaches to develop more robust virtual characters in a dynamically changing environment. Despite the differences in control algorithms, one common focal point of these methods is the improvement of contact control mechanisms. For example, Yin et al.[105] used the position and the velocity of the center of mass to continuously modulate the contact point of the next step. Abe et al. [7] formulated a quadratic program to solve for optimal joint torques subject to frictional contact constraints. Muico et al. [69] developed an online method to adapt the idealized control policy based on the current contact situation. Mordatch et al. [68] planed ground contact positions and the foot trajectory of the center of pressure. Lee et al. [53] showed that robust feedback controller can be achieved by carefully synchronizing the reference trajectory at contact changes. Similarly, in hand animation, Kry and Pai [48] accounted for joint compliance due to contact. Liu [57] optimized the contact positions and forces to synthesize detailed manipulation. Our motivation is similar in that we seek to create more realistic human motion through a better understanding of the contact phenomenon, but we take a drastically different approach. Instead of improving the control algorithm, our method aims to improve the physical realism of contact by simulating the contact points as deformable bodies rather than rigid bodies.

Creating body deformation for an articulated figure is an important and practical problem in computer animation. Common skeleton-driven techniques, such as skeleton-subspace-deformation [64, 1], have been widely adopted by graphics practitioners. These basic methods can be enhanced by adding pose examples [54, 46] or additional degrees of freedom [92, 67]. Data-driven methods based on scanned data [11] or dense motion capture data [70] can also create detailed body deformation driven by skeletal motion. These interpolation-based methods are able to produce visually appealing secondary motion. In contrast, our goal is to investigate the impact of deformations caused by collisions on control strategies for physically simulated characters.

One promising way to achieve realistic deformation at the site of contact is to apply physics-based modeling and simulation of skin layer around the skeleton. Earlier work has used mass-spring systems to synthesize deformable skin wrapped around the kinematic articulated figure [34, 87]. Gourret et al. [34] showed that realistic hand deformation in contact with an elastic object can be computed by a numerical method based on FEM. More recently, Pauly et al. [71] used a quasi-rigid model to simulate small deformations at the site of contact to improve the robustness of contact resolution for point cloud surface representations. Capell et al. [15] introduced a framework for simulating skeleton-driven, elastically deformed characters. Their method used a coarse volumetric finite element mesh to represent the deformation of skin, driven by the underlying skeleton motion. Their results highlighted large secondary motion due to inertia rather than the impact of collision. In addition, the skeletal motion was completely pre-scripted and the effect of skin movement did not affect the skeletal motion.

Our work is most relevant to two methods that consider two-way coupling between the skin and skeleton. Kim and Pollard [44, 45] described an efficient coupled system using meshing embedding with FEM. Their method leverages reduced deformable

models and linear-time algorithms for rigid body dynamics to achieve real-time performance. The primary focus of their work is to develop an interactive user interface to control skeleton-driven deformable characters. In contrast, our work aims to investigate the effect of deformable contact on robustness of the control algorithms. Therefore, we choose to implement a more accurate contact model based on the Linear Complementarity Problem (LCP) formulation instead of the penalty-based contact model used in their work. The computational cost of LCP motivated us to design a more compact representation than FEM for deformable bodies. Galoppo et al. [31] introduced a fast formulation to compute skin displacement due to dynamic interplay with bones. They proposed a very efficient but specialized contact model by decoupling skeleton and skin computations using an approximated mass matrix. We believe with an additional implementation of a more accurate contact model, either of the above methods for simulating deformable bodies will be suitable for our applications.

Modeling soft bodies has also generated significant interest in robotics due to its wide range of potential applications. When a full-body humanoid robot moves in an unknown environment or interacts with humans in an unstructured setting, the motion of the robot must be stable and compliant to protect the robot's structure and ensure humans' safety. Researchers have demonstrated that integrating a compliant sole or shock absorbing materials under a humanoid robot's foot improves the stability of dynamic biped walking [17, 99]. Much research effort has also focused on emulating the compliance of anthropomorphic hands. Unlike rigid linked robots, soft robotic hands can conform better to the manipulated objects , enabling more sophisticated tasks and improving dexterity and robustness [23, 98]. Our work is inspired by this line of robotic research, but we aim to create a unified simulation framework for controlling a variety of human movements.

Apart from virtual control and display, modeling contacts are also important for

haptic interfaces that give force feedback to the user synchronized with the visual display. For manipulation of objects, penalty-based contact methods haven been used since humans have a poor position sense that allows tolerance for penetration into the object. However, there are drawbacks associated such as popping through thin objects, force discontinuities due to multiple surfaces near the penetration etc. Methods such as "virtual proxy" [74] and "god-object" [106] overcome these challenges by tracking a virtual point on the surface on the object along with the actual penetrated point. In addition to computing the feedback forces, is it desired of the haptic interfaces to be "passive", which means that the collisions in the system should only extract energy out of the system. The work of Colgate and Schenkel [19] give the condition of how the real damping of the system is related to the virtual stiffness and damping of the modeled system and the sampling rate of the discrete-time controller. Constantinescu et al. [20] use switching stiffness for handling collisions and contacts to perceptually emphasize rigidity of the manipulated object. They resolve collisions using impulse-based resolution and demonstrate passivity for arbitrarily chosen restitution coefficient. In our work, we use LCP method to resolve the collisions. When the object collides with another for the first time, LCP results in forces that are just enough to reduce the velocity of the contact to zero, thereby reducing the kinetic energy of the system. During sustained contact, the force in the normal direction always does zero work. During slipping, the force to the normal of the surface always acts in the direction opposite to the velocity. These conditions ensure that the contact resolution never adds energy to the system.

5.3 Coupled dynamics

Our representation for human skeleton comprises of articulated rigid bodies, each of which is surrounded by a set of point masses representing the surface of the deformable flesh. In this section, we describe the equations of motion coupling the articulated

rigid bodies and the deformable surface.

5.3.1 Articulated rigid body dynamics

The equations of motion of an articulated rigid body system parametrized by generalized coordinates $\mathbf{r}$ (see Equation 43), are given by:

$$M(\mathbf{r})\ddot{\mathbf{r}} + C(\mathbf{r}, \dot{\mathbf{r}})\dot{\mathbf{r}} + \mathbf{g}(\mathbf{r}) = \boldsymbol{\tau} + J_c(\mathbf{r})^T \mathbf{f}_c \tag{109}$$

where M is the mass matrix, C is the Coriolis matrix, $\mathbf{g}$ are the gravitational forces and $\boldsymbol{\tau}$ are the applied generalized forces. The first six values in $\boldsymbol{\tau}$, which correspond to the global rotation and translation of the system, are zero, resulting in an underactuated system. $\mathbf{f}_c$ and J_c are the contact force and the Jacobian respectively for a single contact point. M, C, $\mathbf{g}$ and J_c are non-linear functions of $\mathbf{r}$ while C is linear in $\dot{\mathbf{r}}$. It is straightforward to add more contacts to the equation; for clarity, we describe our equations with only one contact. Note that in the above equation, the Coriolis term is represented as a multiplication of the matrix C with the velocity vector $\dot{\mathbf{r}}$ whereas in Equation 43, C is simply this product vector. Here, we separate out the matrix C from Equation 43 to emphasize its linear dependence on $\dot{\mathbf{r}}$:

$$C(\mathbf{r}, \dot{\mathbf{r}}) = \sum_k \left(J_k^T M_{ck} \dot{J}_k + J_k^T [\tilde{\boldsymbol{\omega}}_k] M_{ck} J_k \right) \tag{110}$$

The matrices $\dot{J}_k$ and $[\tilde{\boldsymbol{\omega}}_k]$ in the above equation are linear in $\dot{\mathbf{r}}$ making the matrix C linear in $\dot{\mathbf{r}}$.

5.3.2 Deformable body dynamics

We define the surface of a deformable body as a 3D manifold triangle mesh formed by a set of point masses at the vertices. The elastic forces applied on each point mass are modeled as linear spring forces. We measure two kinds of deformations and their corresponding restoring forces at each vertex v_i:

- Vertex deformation. For a vertex v_i, we measure the deformation from its rest position $\bar{\mathbf{x}}_i$ as $\mathbf{x}_i - \bar{\mathbf{x}}_i$. The corresponding restoring spring force with a spring stiffness k_v is written as:

$$\mathbf{f}_{1,i} = -k_v(\mathbf{x}_i - \bar{\mathbf{x}}_i) \tag{111}$$

- Edge deformation. The deformation of the edge e_{ij} connecting two vertices v_i and v_j is given by $(\mathbf{x}_i - \mathbf{x}_j) - (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)$. The corresponding restoring force applied on v_i with a spring stiffness k_e is written as:

$$\mathbf{f}_{2,i} = \sum_{j \in N(i)} -k_e\Big((\mathbf{x}_i - \mathbf{x}_j) - (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)\Big) \tag{112}$$

  where $N(i)$ denotes the subset of vertices of the mesh connected to the vertex v_i via an edge.

The force in Equation 111 attempts to keep each vertex at its rest position while the force in Equation 112 tries to maintain the relative position of the vertex with respect to its neighbors. These forces attempt to keep the mesh in its undeformed state and also penalize translation and rotation of the entire mesh in the defined frame of reference.

We collect the positions of all the vertices in a vector $\mathbf{x} \equiv ({\mathbf{x}_1}^T, \ldots, {\mathbf{x}_N}^T)^T$, where N is the number of vertices in the mesh. The restoring forces defined in Equation 111 and Equation 112 for all the vertices can be represented as the product of a sparse stiffness matrix K_x and the deformation $\mathbf{x} - \bar{\mathbf{x}}$ i.e. $\mathbf{f} = K_x(\mathbf{x} - \bar{\mathbf{x}})$. We add a velocity damping force using a damping matrix $K_{\dot{x}}$. The equations of motion for this linear system are written as:

$$M\ddot{\mathbf{x}} = -K_x(\mathbf{x} - \bar{\mathbf{x}}) - K_{\dot{x}}\dot{\mathbf{x}} - \mathbf{g} \tag{113}$$

where M is the diagonal mass matrix with diagonal entries corresponding to each vertex v_i as m_i and $\mathbf{g}$ are the gravitational forces.

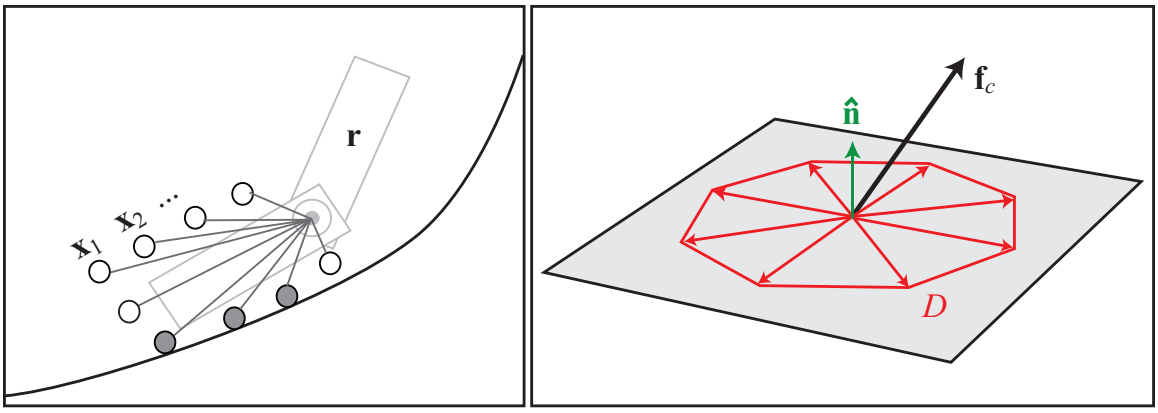


Figure 20: Left: An articulated rigid body system coupled with deformable surface at the site of contact. Solid dots indicate the active DOFs. Right: A contact force $\mathbf{f}_c$ represented by a normal force and a tangent force in coordinates defined by matrix D

5.3.3 Coupled equations of motion

The deformable body dynamics described above are defined for a fixed frame of reference. To drive the deformable body using a rigid body (node) of the articulated skeleton, we need to *attach* the frame of reference of the deformable body to the rigid node. In addition, we must ensure two-way dynamic coupling between the deformable body and the articulated rigid body system.

We augment the articulated body system with point masses corresponding to the vertices of the deformable body (Figure 20, Left). Each point mass is attached to the rigid node through a 3-degree-of-freedom (DOF) translation joint. The DOFs of this joint are the vertex coordinates $\mathbf{x}_i$. By expressing the DOFs of each point mass as a *child* link of the rigid body, we automatically set the frame of reference of the deformable body to the rigid body. This results in an augmented articulated body system that has a tree structure comprising of rigid nodes and point masses. The new set of DOFs or generalized coordinates is represented as $\mathbf{q} \equiv (\mathbf{r}^T, \mathbf{x}^T)^T$. The dynamics for this articulated system is similar to Equation 109 and can be represented as:

$$M\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + \mathbf{g} = \boldsymbol{\tau} + J_c^T \mathbf{f}_c \tag{114}$$

These new matrices M and C and the vector $\mathbf{g}$ can be expressed as:

$$M = \begin{bmatrix} M_r + \bar{M}_x & M_{rx} \\ M_{rx}^T & M_x \end{bmatrix}, C = \begin{bmatrix} C_r + \bar{C}_x & C_{rx} \\ C_{rx}^T & \mathbf{0} \end{bmatrix}, \mathbf{g} = \begin{pmatrix} \mathbf{g}_r + \bar{\mathbf{g}}_x \\ \mathbf{g}_x \end{pmatrix} \tag{115}$$

where M_r and C_r are the mass and Coriolis matrices and $\mathbf{g}_r$ are the gravitation forces defined in Equation 109, M_x is the mass matrix and $\mathbf{g}_x$ are the gravitation forces defined in Equation 113. Equation 115 also introduces a few new coefficients. M_{rx} and C_{rx} are dense matrices coupling the rigid and deformable DOFs and $\bar{M}_x$, $\bar{C}_x$ and $\bar{\mathbf{g}}_x$ are the contributions of the point masses to the dynamic equations of rigid nodes. These matrices can be derived from standard multibody system dynamics. The generalized forces $\boldsymbol{\tau}$ are given by:

$$\boldsymbol{\tau} = \begin{pmatrix} \boldsymbol{\tau}_r \\ \boldsymbol{\tau}_x \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau}_r \\ -K_x(\mathbf{x} - \bar{\mathbf{x}}) - K_{\dot{x}}\dot{\mathbf{x}} \end{pmatrix} \tag{116}$$

where $\boldsymbol{\tau}_r$ are the generalized forces in Equation 109. For an actively controlled articulated system, $\boldsymbol{\tau}_r$ is usually computed by an external controller.

Adaptive deformable simulation. A mesh with large number of vertices makes the dynamic system very expensive to compute. Since we are interested in simulating the deformable body at the site of contact, we only simulate a subset of the vertices at the site of contact and treat the rest of the surface rigid. Intuitively, this implies that we change the stiffness of the rest of the point masses such that they provide the constraint forces required to keep the surface rigid. Now, given the contact points on the surface, we traverse a p-ring neighborhood of the vertices and mark these visited vertices. We simulate those marked vertices until they are no longer in contact and reach the equilibrium at their rest positions. This adaptive scheme for the deformable body simulation is a practical solution for situations involving small and stiff deformations. However, we also need to consider the dynamic contribution of the point masses not simulated, because their effect on the mass matrix in Equation 115

changes over time due to their dependency on the rigid pose $\mathbf{r}$. To efficiently compute the mass matrix, we precompute the contribution of the point masses in their rest positions relative to the local frame of the parent rigid body. This contribution effectively changes the mass and inertia of the parent rigid body. At each time step during the simulation, since we visit all the simulated point masses that are potentially not at their rest positions, we simply subtract their rest pose contribution to the mass matrix from the computed value.

Discrete equations. In our implementation, we discretize the equations of motion in time. Quantities at any time t_k are subscripted by k. For clarity, we time-subscript the quantities only to disambiguate; otherwise they are assumed to be evaluated at time t_k. We use backward and central differences to discretize velocity and acceleration respectively using the time step h as:

$$\begin{aligned} \dot{\mathbf{q}}_k &\equiv \frac{\mathbf{q}_k - \mathbf{q}_{k-1}}{h} \\ \ddot{\mathbf{q}}_k &\equiv \frac{\mathbf{q}_{k+1} - 2\mathbf{q}_k + \mathbf{q}_{k-1}}{h^2} = \frac{\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k}{h} \end{aligned} \tag{117}$$

Substituting the above in Equation 114 and rearranging terms, we arrive at:

$$M(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k) - hJ_c{}^T\mathbf{f}_c = h(\boldsymbol{\tau} - C\dot{\mathbf{q}}_k - \mathbf{g}) \tag{118}$$

Large spring stiffness or damping in Equation 116 may cause instabilities for large time steps. Therefore, we apply the spring forces in an implicit manner i.e. the spring forces are evaluated at time t_{k+1} by using Equation 117 as:

$$\begin{aligned} \hat{\boldsymbol{\tau}}_x &= -K_x(\mathbf{x}_{k+1} - \bar{\mathbf{x}}) - K_{\dot{x}}\dot{\mathbf{x}}_{k+1} \\ &= -K_x(\mathbf{x}_k + h\dot{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}) - K_{\dot{x}}\dot{\mathbf{x}}_{k+1} \\ &= -K_x(\mathbf{x}_k - \bar{\mathbf{x}}) - (hK_x + K_{\dot{x}})\dot{\mathbf{x}}_k - (hK_x + K_{\dot{x}})(\dot{\mathbf{x}}_{k+1} - \dot{\mathbf{x}}_k) \end{aligned} \tag{119}$$

Replacing $\boldsymbol{\tau}_x$ with $\hat{\boldsymbol{\tau}}_x$ in Equation 118 and rearranging terms, we get:

$$\hat{M}(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k) - hJ_c{}^T\mathbf{f}_c = h(\hat{\boldsymbol{\tau}} - C\dot{\mathbf{q}}_k - \mathbf{g}) \tag{120}$$

$$\text{where } \hat{M} = \begin{bmatrix} M_r + \bar{M}_x & M_{rx} \\ M_{rx}^T & M_x + h^2K_x + hK_{\dot{x}} \end{bmatrix}$$

$$\hat{\boldsymbol{\tau}} = \begin{pmatrix} \boldsymbol{\tau}_r \\ -K_x(\mathbf{x}_k - \bar{\mathbf{x}}) - (hK_x + K_{\dot{x}})\dot{\mathbf{x}}_k \end{pmatrix}$$

This semi-implicit system of equations is stable with respect to the deformable forces. Some of other instabilities due to large velocities can be tackled by making the Coriolis term implicit in $\dot{\mathbf{q}}$. From Equation 110, we use the linear dependence of C on $\dot{\mathbf{q}}$ and evaluate $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ using the velocities at the next time step $\dot{\mathbf{q}}_{k+1}$:

$$\begin{aligned} C(\mathbf{q}_k, \dot{\mathbf{q}}_{k+1})\dot{\mathbf{q}}_{k+1} &= C(\mathbf{q}_k, \dot{\mathbf{q}}_k + \Delta\dot{\mathbf{q}})(\dot{\mathbf{q}}_k + \Delta\dot{\mathbf{q}}) \\ &= \left(C(\mathbf{q}_k, \dot{\mathbf{q}}_k) + C(\mathbf{q}_k, \Delta\dot{\mathbf{q}})\right)(\dot{\mathbf{q}}_k + \Delta\dot{\mathbf{q}}) \\ &\approx C(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + C(\mathbf{q}_k, \dot{\mathbf{q}}_k)\Delta\dot{\mathbf{q}} + C(\mathbf{q}_k, \Delta\dot{\mathbf{q}})\dot{\mathbf{q}}_k \\ &= C(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + 2C(\mathbf{q}_k, \dot{\mathbf{q}}_k)(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k) \end{aligned} \tag{121}$$

We substitute this result in Equation 120. The only change in the equation is the mass matrix:

$$\hat{M} = \begin{bmatrix} M_r + \bar{M}_x & M_{rx} \\ M_{rx}^T & M_x + h^2K_x + hK_{\dot{x}} \end{bmatrix} + 2hC$$

Given the current state $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$, the only unknowns in Equation 120 are $\dot{\mathbf{q}}_{k+1}$ in the absence of contact. We can then directly integrate to the next state after we solve for $\dot{\mathbf{q}}_{k+1}$. To incorporate the effect of contact, however, we need to consider additional constraints described in the next section.

5.4 Contact model

Similar to the numerous options for modeling deformable bodies, there are many existing methods for modeling collisions as well. We choose a model that accurately

simulates unilateral contact forces at the points of contact. In addition, we consider the Coulomb friction model that allows for slipping. These conditions for the contact force and the contact point can be formulated into a linear complementarity problem (LCP), as described in [13].

The contact force is represented by its normal and tangential components, $\mathbf{f}_c = f_n\hat{\mathbf{n}} + D\boldsymbol{\beta}$, where $\hat{\mathbf{n}}$ is the normal vector at contact and D is a matrix with its columns as the vectors that span the tangent plane at contact. The contact force is parametrized by a scalar f_n and a vector $\boldsymbol{\beta}$ in the normal and the tangential space (Figure 20, Right).

Let $\mathbf{p}(\mathbf{q})$ be the coordinates of the contact point. The point velocity is linearly related to the generalized velocity as $\dot{\mathbf{p}} = J_c\dot{\mathbf{q}}$. In the normal direction, the projection of the point velocity and the normal force have to satisfy the following condition to prevent penetration of the surface and enforce work-less and unilateral properties of the contact force:

$$\begin{aligned} &(\dot{\mathbf{p}}_{k+1} \cdot \hat{\mathbf{n}}) \perp f_n \\ \text{or } &(\hat{\mathbf{n}}^T J_c \dot{\mathbf{q}}_{k+1}) \perp f_n \end{aligned} \tag{122}$$

where the notation $\mathbf{a} \perp \mathbf{b}$ implies the complementarity condition $\mathbf{a}^T\mathbf{b} = 0$ with $\mathbf{a}, \mathbf{b} \geq \mathbf{0}$. Note that non-negativity ensures component-wise complementarity $a_i \perp b_i$. In the tangential direction, we introduce a parameter λ that represents the relative tangential movement for the point at contact. If there is slipping ($\lambda > 0$), the tangential force should lie on the boundary of the polyhedra in the direction opposite to the movement. If the contact is static, the magnitude of the tangential force should be bound by the normal force modulated by friction coefficient. These two conditions can be enforced by:

$$\begin{aligned} &(\lambda\mathbf{e} + D^T J_c \dot{\mathbf{q}}_{k+1}) \perp \boldsymbol{\beta} \\ \text{and } &(\mu f_n - \mathbf{e}^T\boldsymbol{\beta}) \perp \lambda \end{aligned} \tag{123}$$

where μ is the friction coefficient and $\mathbf{e}$ is a vector of ones. We refer the reader to [84, 13] for a more detailed explanation of the complementarity conditions.

Let $\mathbf{z} = (f_n, \boldsymbol{\beta}^T, \lambda)^T$. Rewriting Equation 120 with the parametrized contact force, we get:

$$\begin{bmatrix} \hat{M} & -(\hat{\mathbf{n}}^T J_c)^T & -(D^T J_c)^T & 0 \end{bmatrix} \begin{pmatrix} \dot{\mathbf{q}}_{k+1} \\ \mathbf{z} \end{pmatrix} + \mathbf{b} = \mathbf{0} \tag{124}$$

where $\mathbf{b} = -\hat{M}\dot{\mathbf{q}}_k - h(\hat{\boldsymbol{\tau}} - C\dot{\mathbf{q}}_k - \mathbf{g})$. Combining the above constraint with the complementarity conditions in Equation 122 and Equation 123, we get:

$$\begin{bmatrix} \hat{M} & -(\hat{\mathbf{n}}^T J_c)^T & -(D^T J_c)^T & 0 \\ \hat{\mathbf{n}}^T J_c & 0 & \mathbf{0} & 0 \\ D^T J_c & 0 & \mathbf{0} & \mathbf{e} \\ 0 & \mu & -\mathbf{e}^T & 0 \end{bmatrix} \begin{pmatrix} \dot{\mathbf{q}}_{k+1} \\ \mathbf{z} \end{pmatrix} + \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{w} \end{pmatrix}$$
$$\text{with } \mathbf{w} \perp \mathbf{z} \tag{125}$$

Equation 125 represents a mixed LCP problem with $\mathbf{w}, \mathbf{z}$ and $\dot{\mathbf{q}}_{k+1}$ as unknowns. Given the current state of the character $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$ and the contact points, we solve this problem to get the generalized velocities $\dot{\mathbf{q}}_{k+1}$ for the next time step t_{k+1} and compute the new state as $(\mathbf{q}_k + h\dot{\mathbf{q}}_{k+1}, \dot{\mathbf{q}}_{k+1})$.

5.5 Implementation of controllers

Without active control forces $\boldsymbol{\tau}_r$ applied at the actuated joints of the skeleton, the system described in previous sections is completely passive. To demonstrate the utility of our coupled dynamic system, we implement the following control algorithms based on existing controllers widely adopted in computer animation and robotics. Their applications in locomotion and manipulation are demonstrated in Section 5.6.

5.5.1 Pose-space tracking control

We formulate an implicit PD control for tracking a given pose or a sequence of poses for the articulated skeleton. Let $(\mathbf{r}_k, \dot{\mathbf{r}}_k)$ denote the current state for the rigid skeleton and $\bar{\mathbf{r}}$ be the desired pose. Further, let K_r and $K_{\dot{r}}$ be diagonal stiffness and damping matrices respectively. To provide more stability in the system, we calculate the feedback force based on the deviation of the *next* state from the desired pose, similar to Equation 119:

$$\begin{aligned}\boldsymbol{\tau}_r =& -K_r(\mathbf{r}_{k+1} - \bar{\mathbf{r}}) - K_{\dot{r}}\dot{\mathbf{r}}_{k+1} \\ =& -K_r(\mathbf{r}_k - \bar{\mathbf{r}}) - (hK_r + K_{\dot{r}})\dot{\mathbf{r}}_k - (hK_r + K_{\dot{r}})(\dot{\mathbf{r}}_{k+1} - \dot{\mathbf{r}}_k)\end{aligned} \tag{126}$$

Plugging $\boldsymbol{\tau}_r$ into Equation 120 and rearranging the equation, we modify $\hat{M}$ and $\hat{\boldsymbol{\tau}}$ in Equation 120 as:

$$\begin{aligned}\hat{M} &= \begin{bmatrix} M_r + \bar{M}_x + h^2K_r + hK_{\dot{r}} & M_{rx} \\ M_{rx}^T & M_x + h^2K_x + hK_{\dot{x}} \end{bmatrix} + 2hC \\ \hat{\boldsymbol{\tau}} &= \begin{pmatrix} -K_r(\mathbf{r}_k - \bar{\mathbf{r}}) - (hK_r + K_{\dot{r}})\dot{\mathbf{r}}_k \\ -K_x(\mathbf{x}_k - \bar{\mathbf{x}}) - (hK_x + K_{\dot{x}})\dot{\mathbf{x}}_k \end{pmatrix}\end{aligned} \tag{127}$$

5.5.2 Cartesian-space tracking control

Sometimes it is more effective to track a position in Cartesian-space rather than in the pose space. Let $\mathbf{p}_i(\mathbf{r})$ represent the world coordinates of a point on a rigid body. Let the desired position of this point be $\bar{\mathbf{p}}_i$. For small deviations, we can approximate it as $\Delta\mathbf{p}_i \equiv \mathbf{p}_i - \bar{\mathbf{p}}_i \approx J_i\Delta\mathbf{r}$, where J_i is the Jacobian evaluated at $\mathbf{p}_i$. With this approximation, we can simply use the pose-space PD control as described in Equation 127 to track the new desired pose as $\bar{\mathbf{r}} + \Delta\mathbf{r}$. The same approximation can be applied to tracking multiple Cartesian points:

$$\begin{bmatrix} J_1^T, & \dots & , J_m^T \end{bmatrix}^T \Delta\mathbf{r} = \begin{pmatrix} \Delta\mathbf{p}_1^T, & \dots & , \Delta\mathbf{p}_m^T \end{pmatrix}^T \tag{128}$$

Similarly, body orientation in Cartesian-space can be controlled using the same mechanism: $\Delta\omega_b = J_{\omega_b}\Delta\mathbf{r}$, where $\Delta\omega_b$ is the change in orientation of the body and J_{ω_b} is the Jacobian defined as $\omega_b = J_{\omega_b}\dot{\mathbf{r}}$.

5.5.3 Locomotion control

In addition to basic tracking controllers, we also apply our coupled dynamic system to a biped controller, SIMBICON [105], which has been adopted widely by other researchers in computer graphics community [22, 21, 90, 91, 53].

SIMBICON uses maximal coordinates to compute joint torques and employs Open Dynamics Engine [5] to solve for the simulation time step. The algorithm for advancing one time step in original SIMBICON can be summarized as:

STEP 1: Compute joint torques $\boldsymbol{\tau}_s$

STEP 2: Detect collisions

STEP 3: Create contacts to be solved for in ODE

STEP 4: Apply $\boldsymbol{\tau}_s$ to character in ODE

STEP 5: Advance one time step in ODE to get next state

Our method follows the SIMBICON algorithm except for the contact force handling. To compute the contact forces for our coupled dynamic system, we first need to convert the torques and the state from maximal coordinates used in SIMBICON to our generalized coordinates. We then solve for contact forces via Equation 125 and use them to override the contact forces solved by ODE. Our contact handling, summarized as follows, replaces the STEP 3 of SIMBICON algorithm.

STEP 3.1 Convert $\boldsymbol{\tau}_s$ to generalized torques $\boldsymbol{\tau}_r$ (see Section 2.4.2)

STEP 3.2 Convert state to generalized coordinates $(\mathbf{r}_k, \dot{\mathbf{r}}_k)$ (see Section 2.4.1)

STEP 3.3 Solve $(\mathbf{q}_{k+1}, \dot{\mathbf{q}}_{k+1})$ and $\mathbf{f}_c$ using Equation 125

STEP 3.4 Apply $\mathbf{f}_c$ to character in ODE

Performance improvement. Although our contact handling for the coupled dynamic system inevitably increases the computation time, our semi-implicit scheme in $\hat{\boldsymbol{\tau}}$ potentially allows for a larger time step than the one used by SIMBICON (0.5ms), resulting in less frequent computation of Equation 125. However, using a smaller frequency for the contact handling poses problems to the rest of the simulation algorithm; infrequent updates of the deformable state and the contact forces can lead to inaccurate collision detection and dynamic inconsistency. To address this problem of asynchronous time updates, we introduce a mixed-frequency simulation algorithm.

Let h_s be the SIMBICON time step used in STEP 5 and $h_d = nh_s$ be the time step for solving Equation 125 in STEP 3.3 for some integer n. At time t_0, we execute STEP 3.1 to STEP 3.4; i.e. we solve Equation 125 and record the contact forces $\mathbf{f}_c$ and the new deformable state $(\mathbf{x}_{k+1}, \dot{\mathbf{x}}_{k+1})$. The new deformable state and contact forces are applied at time $t_0 + h_d$. For any time $t \in (t_0, t_0 + h_d)$, we interpolate the deformable state as $(\mathbf{x}_t, \dot{\mathbf{x}}_t) = (1-u)(\mathbf{x}_k, \dot{\mathbf{x}}_k) + u(\mathbf{x}_{k+1}, \dot{\mathbf{x}}_{k+1})$ where $u = (t - t_0)/h_d$. For the contact force during $t \in (t_0, t_0 + h_d)$, we could simply treat the deformable as a rigid body and use STEP 3 to compute the contact forces. However, this treatment largely reduces the overall effect of the coupled dynamic system. Instead, we take into account the impact of deformable body on the rigid DOFs as an *additional generalized force*, $\bar{\boldsymbol{\tau}}_x$, applied to the character before solving the ODE time update:

$$\bar{\boldsymbol{\tau}}_x = -(\bar{M}_x\ddot{\mathbf{r}} + M_{rx}\ddot{\mathbf{x}} + \bar{C}_x\dot{\mathbf{r}} + C_{rx}\dot{\mathbf{x}} + \bar{\mathbf{g}}_x) \tag{129}$$

Equation 129 can be derived from Equation 114 and Equation 115:

$$\begin{aligned} &(M_r + \bar{M}_x)\ddot{\mathbf{r}} + M_{rx}\ddot{\mathbf{x}} + (C_r + \bar{C}_x)\dot{\mathbf{r}} + C_{rx}\dot{\mathbf{x}} + (\mathbf{g}_r + \bar{\mathbf{g}}_x) = \boldsymbol{\tau}_r + J_c^T\mathbf{f}_c \\ \Rightarrow\ &M_r\ddot{\mathbf{r}} + C_r\dot{\mathbf{r}} + \mathbf{g}_r = \boldsymbol{\tau}_r + J_c^T\mathbf{f}_c - (\bar{M}_x\ddot{\mathbf{r}} + M_{rx}\ddot{\mathbf{x}} + \bar{C}_x\dot{\mathbf{r}} + C_{rx}\dot{\mathbf{x}} + \bar{\mathbf{g}}_x) \end{aligned}$$

Leaving out the last term in the RHS (i.e. $\bar{\boldsymbol{\tau}}_x$), the above equation represents the equations of motion of an articulated rigid body system with DOFs $\mathbf{r}$ (Equation 109). The accelerations $\ddot{\mathbf{r}}$ and $\ddot{\mathbf{x}}$ for the duration $(t_0, t_0 + h_d)$ can be computed using their

discretizations (Equation 117) since we have both the velocities, $\dot{\mathbf{q}}_k$ and $\dot{\mathbf{q}}_{k+1}$ at times t_0 and $t_0 + h_d$ respectively.

Stability improvement. Because the applied torques $\boldsymbol{\tau}_s$ computed by SIMBICON are based on a very small time step, directly applying them to our formulation with a much larger time step can sometimes lead to instability. To counter this, we add one term to the converted generalized torques $\boldsymbol{\tau}_r$ to approximate the effect of an implicit integration scheme. This modified $\boldsymbol{\tau}_r$ are only used to compute more stable contact forces in STEP 3.3, and we still use the original $\boldsymbol{\tau}_s$ in ODE forward simulation (STEP 4). We define a vector $\mathbf{k}_\tau$ such that $\boldsymbol{\tau}_r = K_\tau(\mathbf{r}_k + h_d\dot{\mathbf{r}}_\mathbf{k}) + \mathbf{k}_\tau$ for some chosen positive definite matrix K_τ. We now make an approximation by replacing $\dot{\mathbf{r}}_k$ with $\dot{\mathbf{r}}_{k+1}$ thus making the forces implicit since $\dot{\mathbf{r}}_{k+1}$ is unknown: $\boldsymbol{\tau}_r \approx K_\tau(\mathbf{r}_k + h_d\dot{\mathbf{r}}_{\mathbf{k+1}}) + \mathbf{k}_\tau$. The resulting deviation added to $\boldsymbol{\tau}_r$ equals $h_d K_\tau(\dot{\mathbf{r}}_{\mathbf{k+1}} - \dot{\mathbf{r}}_\mathbf{k})$ or $h_d^2 K_\tau \ddot{\mathbf{r}}$. Introducing this additional term produces more stable contact forces without altering the original torques generated by the SIMBICON controller.

5.6 Results

We tested results on controllers for biped locomotion and hand manipulation. All the results were produced on a single core of 2.7 GHz Intel i7. For each behavior, we applied the identical control algorithm to a character with soft tissue at the site of contact (soft character) and a character comprising only rigid bodies (rigid character). We compared the motions of these two characters.

Biped locomotion controller. Using the SIMBICON-based controller described in the previous section, we designed three experiments to evaluate the impact of soft contacts. We directly used the source code of SIMBICON and ODE for the control force computation and the forward simulation. The only modification by our method was the contact force handling.

In the first experiment, we applied large push forces to the character and compared the motion with and without soft contact simulation. SIMBICON is known for its robustness to external perturbations. However, when a strong push that induces a large torque, the rigid character tends to lose contact easily and fails to recover. In contrast, the deformable bodies allow our character to maintain more contact points on the ground with more evenly distributed pressure (Figure 21). Losing a few contact points due to the perturbation does not critically affect the balance state.

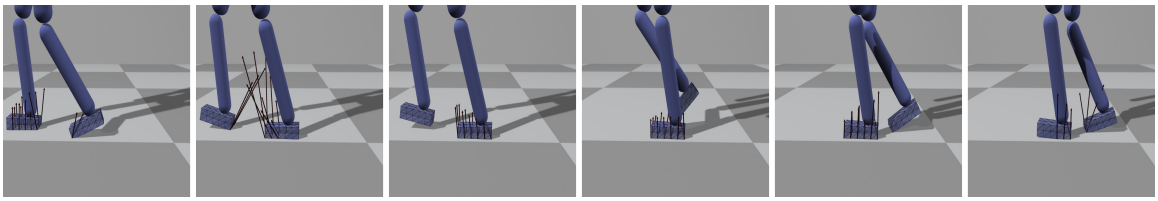


Figure 21: Simulating deformable body at the site of contact results in more contact points and smoother center of pressure.

The second and third experiments focus on evaluating the controller under different sources of uncertainty. We first considered the noise in the motor control system of the character. We implemented the same simplified, biologically-inspired model as Wang et al. [91]. This model adds noise to the joint torques produced by the controller at every time step. The noise is drawn from a Gaussian density with zero mean and a standard deviation depending on the magnitude of the joint torque and the strength of the joint. We used middle noise level (i.e. $\beta = 75$ defined in their paper) to test our controllers. As a result, the rigid character quickly becomes unstable when small pushes are applied. The soft character, on the other hand, is still able to maintain balance and withstand large pushes.

The last experiment evaluates the biped controller operating on a noisy surface. We first segmented the floor into small tiles of 5×5 cm^2. For each vertex, we added a random offset, uniformly sampled from a range of [0,2] cm, to its vertical and horizontal positions. We then reconstructed a bumpy surface based on the modified

vertices. The controllers were tested on several randomly generated different floors from which we demonstrate three. The uncertainty on the surface greatly affects the rigid character's ability to walk. In some cases it wanders to various locations and other cases it simply falls. The soft character is able to stay in the original course for all cases with small variations. A close-up observation shows that the deformable foot mesh of the soft character conforms better to the noisy surface and maintains more contact points when the character is pushed.

Cartesian-space tracking controller. We designed a manipulation controller based on tracking the center of pressure on the finger. The intended function of the controller is to flick a marble ball in the desired direction using the distal phalanx of the index finger and the thumb. The controller attempts to match the relative center of pressure between the thumb and the index finger to a desired vector, in addition to tracking an equilibrium pose.

The rigid hand demonstrates no control over the direction of the ball. In some cases, it fails to launch the ball because the loss of contact points occurs too early. The soft hand shows much more accurate control in different directions and the ball never slips off the fingers before the launch. We also tested the robustness of the controller by adding noise to the surface of the ball. By applying the same control forces several times, the fingers with soft contact manage to flick the ball in the similar directions, but the rigid fingers produce motions with huge variance. The key difference that leads to better dexterity is that the soft hand maintains more contact points and smoother movement of the center of pressure at all times.

Pose-space tracking controller. We tested pose-space tracking control strategy on a human upper body and a human hand. For the human upper body, our goal is to track an arm-folding pose (Figure 22). Although it is not a difficult control problem, this particular pose is very difficult to simulate due to a large area of contact against

multiple body parts. In our results, the rigid character immediately gets stuck when the hand collides into the opposite upper arm. In contrast, the soft character is able to fold her arms with the hands and upper arms brushing against each other and smoothly moving into their own desired positions.

Figure 22: Comparison of the arm folding pose between a "rigid character" (Left) and a "soft character" (Right).

We also developed a controller capable of pinch-grasping a thin object. We employed PD controllers to track an equilibrium pose such that a pen can be held horizontally in between the index finger and the thumb (Figure 23). Without any perturbation, both the rigid hand and the soft hand can successfully hold the pen. However, when external forces are applied to the pen, the rigid hand quickly loses contacts and drops the pen while the soft hand can withstand perturbation in any direction. The total contact forces applied on the pen are comparable between two hands, but the pressure distribution is much smoother on the soft hand.

We summarize the performance and the parameters used in our simulations in Table 1. The stiffness parameter k_m is the representative stiffness for k_v and k_e in Equation 111 and Equation 112. We define the values $k_v = 0.4k_m$ and $k_e = 0.6k_m$ for all our examples. In addition, we define the damping matrix $K_{\dot{x}} = 0.1k_m\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. For the locomotion examples, we define $K_\tau = 10^3$ (Section 5.5.3). The bottleneck in the computation is the number of contacts

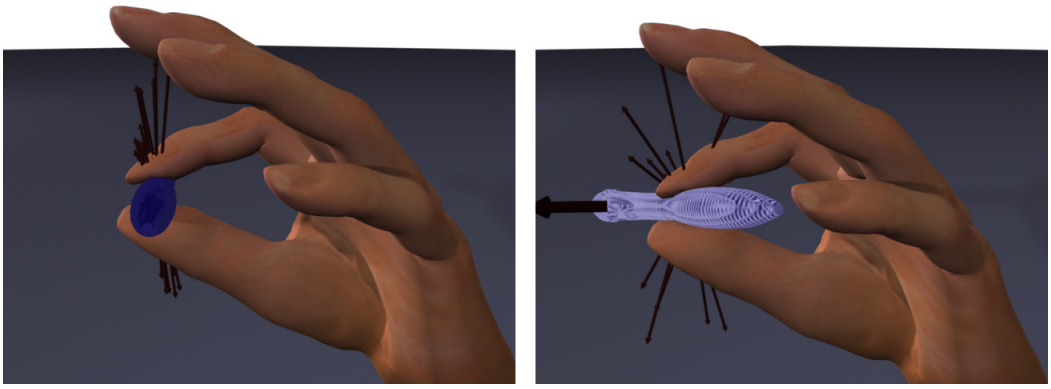


Figure 23: Soft fingers enable a more robust pinch-grasp.

Table 1: Performance and parameters of the examples. "total DOFs" is the number of DOFs that can be potentially simulated while "simulated DOFs" is the number of DOFs in adaptive simulation. "fps" is the frame rate for our simulations and "LCP time" is the percentage of the simulation time to solve Equation 125. For biped walk, the LCP is solved every 8 SIMBICON time steps (SIMBICON time step is 0.5 ms).

	total DOFs	simulated DOFs	num contacts	fps	LCP time (%)	Time step (ms)	Stiffness k_m
finger flick	2573	576±88	39±6	3.9±3.2	86±6	1.7	1.5×10^4
arm fold	2802	322±89	33±10	3.5±1.7	68±10	8.3	10^4
pinch grasp	1427	258±22	29±4	5.2±3.2	85±6	1.7	1.5×10^4
biped walk	334	197±43	16±3	18.5±4.5	63±5	4.0	10^3

solved by LCP for every time step. We use the publicly available PATH LCP solver (http://www.cs.wisc.edu/cpnet/cpnetsoftware/) to solve for Equation 125 for both the cases of rigid and soft contacts.

5.7 Discussion

In addition to the comparisons described in Section 5.6, we performed quantitative analyses to show the impact of soft contact, the effect of a few key parameters, and comparisons with alternative design decisions. We also address a few limitations of the current system in this section.

5.7.1 Evaluation

Center of pressure. We plotted the center of pressure of the character's foot from one of the biped examples (Figure 24). The comparison shows that a soft contact has a much smoother center of pressure than the rigid contact. Because the center of pressure is crucial to maintaining the angular momentum, this result is consistent with our observation that the rigid character loses balance more easily when large external torque is applied. Similarly, we compared the magnitude of the total contact force from the same example (Figure 25). Open Dynamics Engine (ODE) uses the *constraint force mixing (cfm)* parameter to control the "softness" of the constraints. In order to compare our soft contacts with the softer constraints at the contacts points for rigid body collisions in ODE, we increase the *cfm* parameter by 10 and 20 times respectively. SIMBICON uses $cfm = 10^{-5}$. Figure 26 and Figure 27 compare the force magnitude and the center of pressure during a step between the soft contacts and softer ODE contacts. The average contact force is usually similar between a rigid and a soft contact, but the rigid contact has a larger variance in the magnitude of contact force. Note that more sophisticated biped controllers might achieve similar results, but we deliberately choose very simple control algorithms to highlight the effect of soft body contact.

Contribution to robustness. The main reason that soft bodies can generate contact forces resulting in a more robust motion is due to the increase in the number of contact points. In all our examples, the number of contacts of soft bodies is significantly greater than that of rigid bodies. For example, we compared the number of contact points of rigid and soft fingers in a pinch-grasp motion (Figure 23). Figure 28 suggests that soft fingers maintain a large number of contacts at all times and do not abruptly change the number of contacts under external forces. In contrast, the number of contacts on rigid fingers is much smaller and fluctuates drastically when

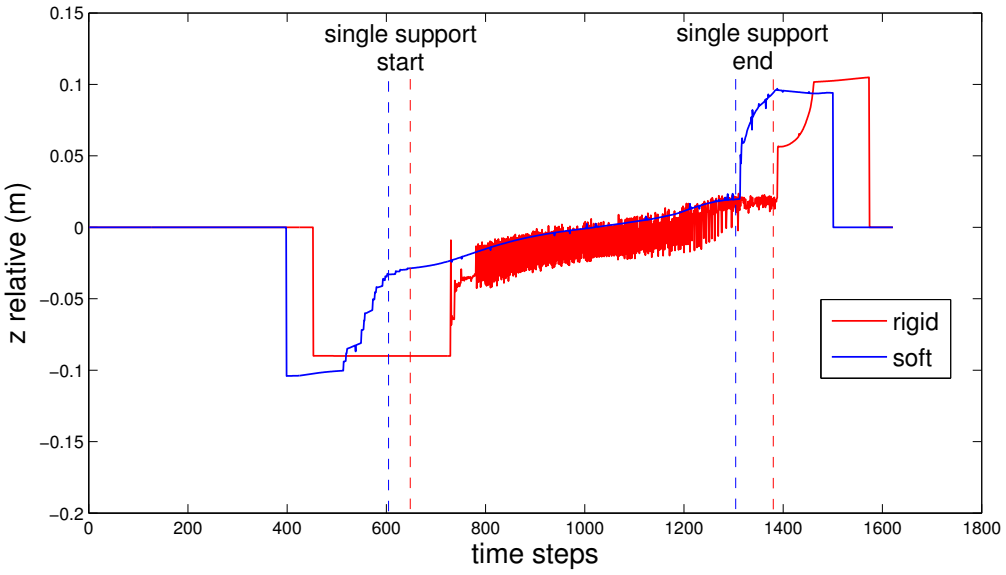


Figure 24: Comparison of the center of pressure on the left foot in the direction of heel to toe for one step (frames 400-1500)

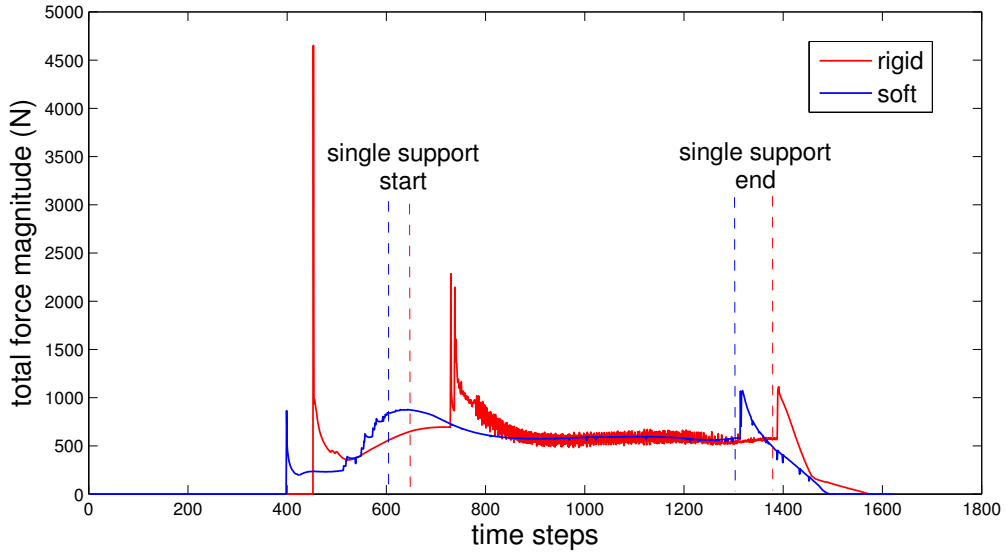


Figure 25: Comparison of the magnitude of the total contact force on the left foot for one step (frames 400-1500).

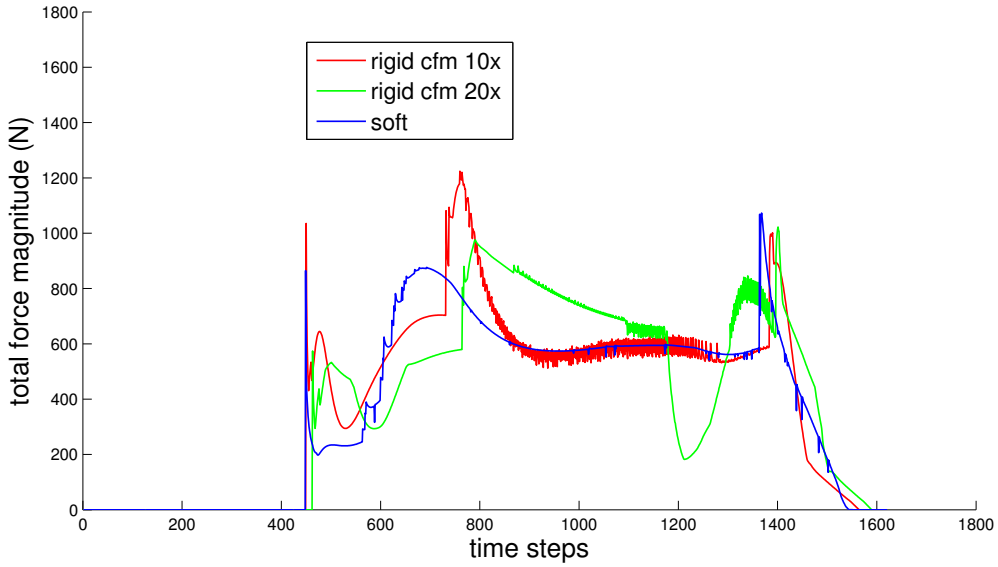


Figure 26: Comparison of the magnitude of the total contact force on the left foot for one step (frames 400-1500). The red and green plots correspond to the softer contact constraints in ODE.

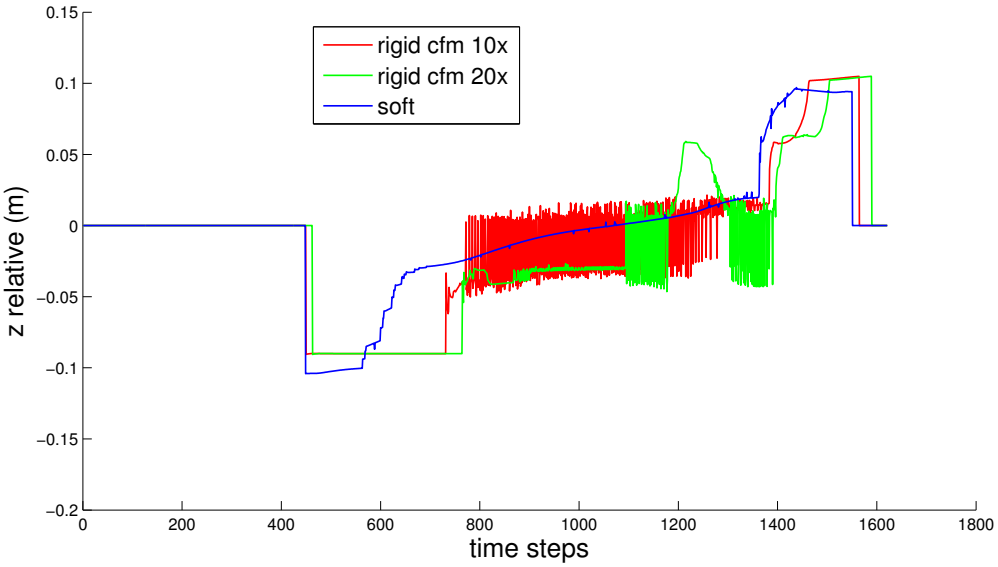


Figure 27: Comparison of the center of pressure on the left foot in the direction of heel to toe for one step (frames 400-1500). The red and green plots correspond to the softer contact constraints in ODE.

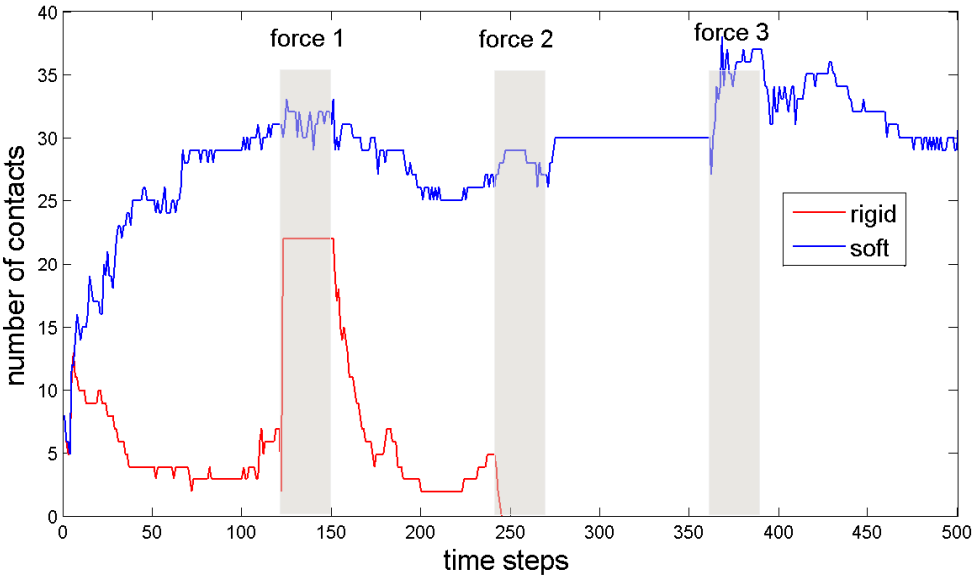


Figure 28: Comparison of the number of contacts from the pinch-grasp example.

an external force is applied, leading to unpredictable and less stable results. We also verified that the robustness of the motion was not due to the implicit formulation. In this experiment, we replaced all the implicit formulations with explicit ones and simulated the same biped examples with a smaller time step. Both implicit and explicit characters recover from the push in a similar manner.

System parameters. Our framework performs robustly for a wide range of parameters such as the stiffness of the mesh and time steps. We use a time step of 4ms for locomotion controllers (see Table 1). We also performed experiments with an larger time step of 8ms. The resulting controller was still robust to large forces as compared to the original SIMBICON controller. In addition, we experimented with stiffness of the foot meshes increased by a factor of 10-20 and the controller performed robustly.

Different LCP solutions. In theory, the mixed LCP problem in Equation 125 can have multiple solutions depending on the initial point given to the PATH solver. In practice, we empirically showed that our particular LCP, i.e. the contact force

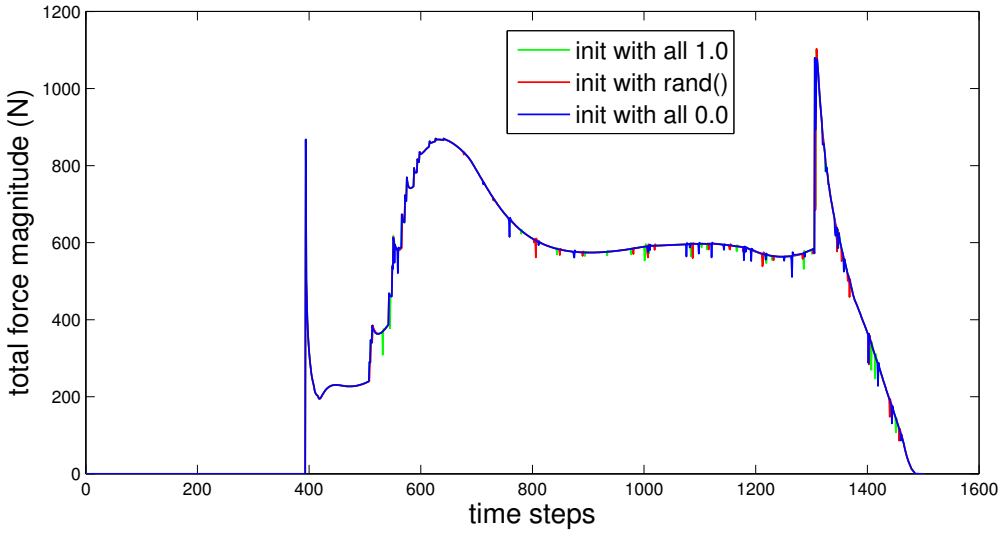


Figure 29: Comparison of the magnitude of contact forces for simulations with different initial points for the LCP solver.

problem, is not sensitive to the initial point. We repeatedly simulated the same motion in Figure 21 by solving the LCP with different initial points: a vector with all zeros, a vector with all ones, and a vector of different random values at every frame. Figure 29 shows little variation in the force magnitudes and the simulated motions are close to identical.

Flexible rigid foot vs. deformable foot. Since the compliance near the contact comprises of joint level compliance and the surface compliance, one possible approach to generate robust locomotion is to represent the foot with many rigid links, allowing for more flexible foot motion. We therefore increased the complexity of the rigid foot to four links, each of which is connected to its parent link by a hinge joint. The simulations suggest that a rigid foot represented by four links results in more robust motion when comparing with a rigid foot with fewer links. However, our deformable feet still exhibit more stability than the four-linked rigid feet due to more continuous changes of contacts. In addition, modeling the feet with multiple links increases the complexity of controller design, as more parameters need to be tuned for the

additional actuators on the foot. Increase in the number rigid links that are small as well, the disparity in the maximum and the minimum mass in the system increases leading to numerical instabilities forcing further smaller time steps.

Penalty methods. Penalty-based methods are commonly used to approximate the compliance at the site of contact due to their simple formulation and less intensive computation. However, our approach is advantageous due to the following reasons. First, LCP formulation models more accurate contact by enforcing work-less normal force, no penetration, and realistic slipping. Second, our method explicitly deforms the geometry of the body parts, introducing more contact points which, in turn, increase the robustness of the control algorithms. Third, using low stiffness for penalty methods may lead to frequent penetration artifacts. For a detailed, contact-rich motion, such as hand manipulation, the artifacts can be very visible. We compared the penalty method with our soft contacts for a simple case of a box falling on the ground (see Figure 30). We modeled the penalty forces in the normal direction as described in [101] and vary the stiffness and damping parameters to model different levels of softness and the duration of contact. For the deformable box with soft contacts, we chose the mesh stiffness such that the duration of contact is similar to the corresponding penalty method case. When the stiffness of the box is low, the contact force profiles look similar for both cases in Figure 30. However, the motion generated by the penalty method shows large penetration between the box and the ground while our approach does not. To reduce the penetration, we increase the stiffness of the box; however, it results in larger forces as compared to the LCP method since the latter correctly generates just enough force to stop the hard box and produces no extra bounce. Using high damping in conjunction with high stiffness for penalty methods is a solution; however larger stiffness and damping parameters imposes stricter time step restrictions in the discrete simulation. To have compliance in the contacts, it is

necessary to have less stiffer forces that would lead to some penetration. In addition, simulation would be physically different as compared to our soft contacts method since the soft contacts allows the geometry to change and conform to the surface.

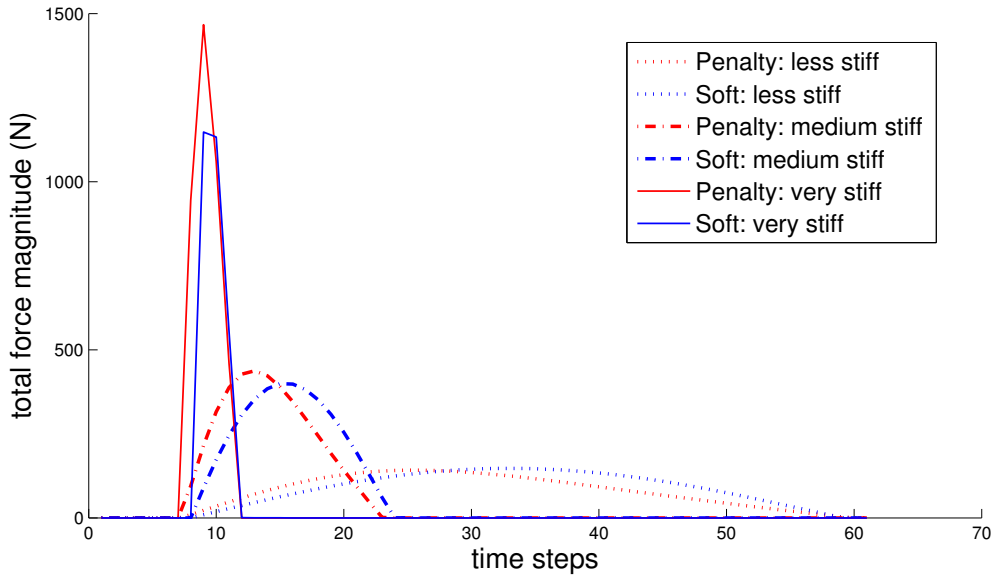


Figure 30: Comparison of the contact force magnitudes from penalty method and soft contacts with LCP by varying the stiffness parameter while keeping the damping constant.

To understand the effect of the stiffness and damping parameters for penalty method and our soft contacts method, we plot the penetration depths and the ratios of the kinetic energy after and before the collision by varying the stiffness and the damping parameters one at a time. Figure 31 shows that the stiffness parameter is responsible for reducing the penetration depth and the damping parameter is responsible to damp out the kinetic energy after the collision. Note that the damping forces start to dominate with the increasing damping parameter and eventually the penetration depth starts to increase. Figure 32 shows a similar comparison for the case of soft contacts that use LCP for contact resolution. Since the LCP method never leads to penetration, both the stiffness and the damping parameters can be used to control the kinetic energy after the contact. This gives an extra degree-of-freedom

to the soft contact formulation that can control the amount of geometry deformation along with the desired bounciness of the contact using these two parameters without leading to any penetration.

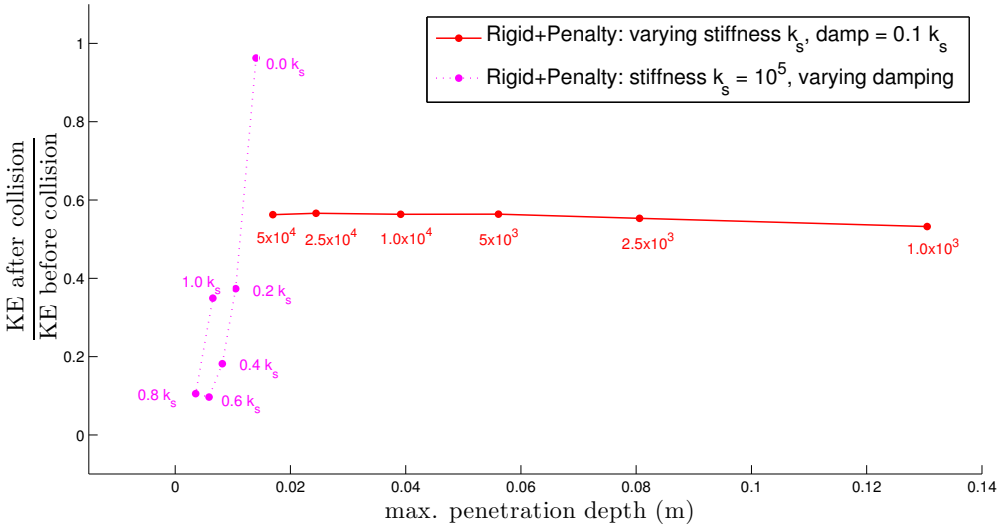


Figure 31: Comparison of the effect of the stiffness and the damping parameters in the penalty method on the maximum penetration depth during the collision and the ratio of kinetic energy preserved after the collision.

Deformation of the body that stores and releases potential energy during a collision can be observed using our approach. This property allows us to control the bounce of the object after collision mimicking the coefficient of restitution (without any penetration). Figure 33 shows how the time of contact and the kinetic energy changes with the stiffness k_m of the deformable box. The softer box is able to store more potential energy which gets converted back to the kinetic energy after the box leaves the ground.

5.7.2 Limitations

There are a few limitations in the current implementation of the algorithm. Our implementation does not handle a deformable body across multiple rigid bones. For motions with large areas of contact, such as rolling on the floor or sitting on a chair, it

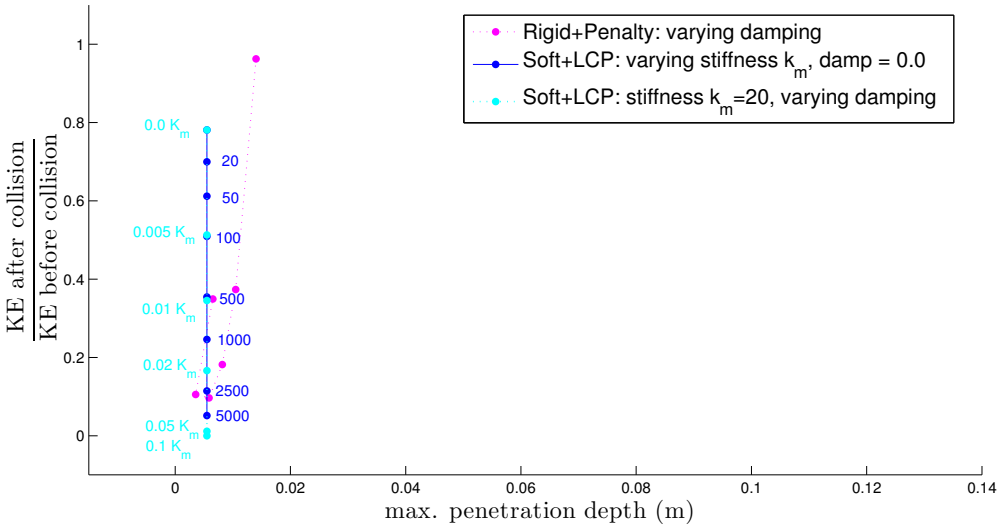


Figure 32: Comparison of the effect of the stiffness and the damping parameters in the soft contacts method on the penetration depth during the collision and the ratio of kinetic energy preserved after the collision. The penetration plot for the penalty method with varying damping in Figure 31 is shown here for reference.

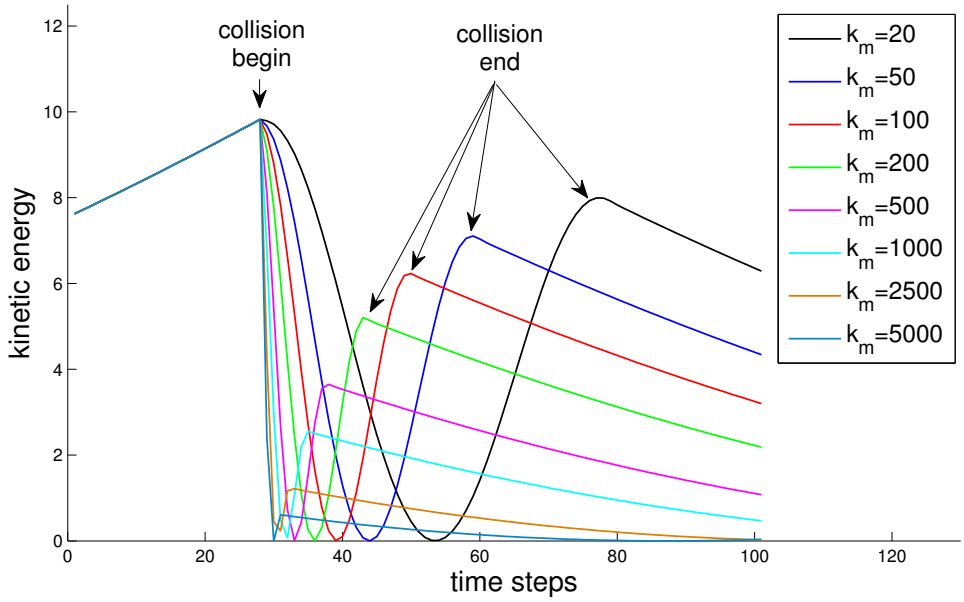


Figure 33: Comparison of responses of the deformable box with different stiffness to the collision with the ground while using the LCP method for contact resolution.

is essential to allow the contact to move continuously across the bones. With moderate effort to convert between frames of reference, we can modify our implementation such

that two connected point masses can be parented to different rigid bones.

Our algorithm is not optimized for the performance. The computation of a few components, such as collision detection and handling can be largely reduced by decoupling the surface mesh resolution and the number of contacts such as using volume contacts as described in a recent work [10]. In addition we can incorporate existing, efficient methods that simulate a fine surface mesh embedded in a coarse control mesh to improve the performance [45].

We have noticed some self-intersection between legs in the biped examples. Our algorithm can handle collision between two body parts at the deformable layer, but when the intersection is deep in the bone level, our algorithm does not handle the collision. This type of deep self-intersection is usually due to the control force generated by a controller that does not take into account self collision. Having a better collision detection routine can improve the situation (we use ODE's build-in collision detection routine), but a more effective solution is probably to design a controller that considers "proprioception" when computing the control forces.

CHAPTER VI

CONCLUSION

In this dissertation, we have presented methods for efficient and robust control of virtual characters in dynamic environments. The main approach of our work is to use the contact configuration of the character with her environment in the control design and solve for both the pose and the contact forces together. We emphasize the impact of the contacts on various control methods by demonstrating superior robustness of the controllers when they employ the soft contact model as compared to the standard rigid body contact model.

In Chapter 3, we described a new approach to synthesize reactive virtual characters in a physical environment based on constrained optimization. Our approach provides a generic framework for rapidly designing a variety of controllers by formulating high-level objectives and tasks in terms of character's pose and contact configuration. This approach gives us the following advantages:

- Our goal-driven formulation of control strategies expedites the design of physics-based motion controllers, enabling the programmer to rapidly create a wide range of motion repertoires for virtual characters.

- The controllers designed in this framework can be robustly adapted to different virtual characters (e.g. adult or child characters) and environments (e.g. slippery surface, cable car environment etc.).

The above approach does not use any motion capture data for guiding the motion synthesis process. Captured motion of a real human provides guidelines for synthesizing natural intentional motion that results from skills acquired by humans over time.

Therefore to successfully synthesize such motions, longer term planning of control is often required during the simulation. In Chapter 4, we presented an algorithm to interactively simulate and control an under-actuated articulated character performing a given reference motion and its variations. We applied modal analysis to transform the space of DOFs to the modal space based on the natural frequencies. The design of control scheme exploits modal analysis to reduce the control space and decouple the equations of motion. In addition, the input motion sequence provides the reference joint angle trajectories along with the contact configuration of the character for the entire motion. This approach offers the following key advantages:

- Robust control: Long-horizon optimization produces look-ahead control that allows the under-actuated character to operate robustly when tracking the reference motion and recovering from small perturbations.

- Online editing of reference trajectory: Because the look-ahead control is computed online at every time step, our algorithm does not rely on any offline computation for control policy. This allows us to edit the reference trajectory online in response to large changes in the environment.

We demonstrated the ability to synthesize anticipatory motion and passive responses to external perturbations. The control policy is generic as it does not depend on the performed task. In addition, chosen weights and parameters work with a wide variety of motions and scenarios.

We presented novel control methods in Chapter 3 and Chapter 4 that use the contact configuration of the character in computing robust contact forces. Rather than focusing on designing new control algorithms, we investigated the impact of *soft contacts* on the existing control algorithms in Chapter 5. We developed a compact soft contact model based on deforming surfaces at the sites of contact. The primary contribution in Chapter 5 is demonstrating that simple control strategies coupled

with the simulation of soft tissue deformation at the site of contact can achieve very robust and realistic motion. We developed a practical system that simulates two-way coupling of rigid and deformable bodies efficiently and robustly. We then applied a few simple and widely used control algorithms, such as pose-space tracking control, Cartesian-space tracking control, and SIMBICON to simulate a variety of behaviors for both full-body locomotion and hand manipulation. We compared the results with motions of a character comprising only of rigid bodies and demonstrated that soft contacts help the control algorithms perform more robustly in comparison to the standard rigid body contacts.

Within this dissertation, we have asserted that the problem of determining control forces for successful execution of any desired behavior is closely tied to the contact configuration and the way these contacts are handled by the control algorithm. This leads to a few interesting avenues for future research.

Biped controllers such as found in [24, 69, 103] do not not use PD feedback control. These controllers are usually more "aware" of contact situations when planning the control forces. We believe that the simulation of deformable bodies at the site of contact could have a great impact on these controllers.

Recent advent in biped controllers primarily focused on robust locomotion. Motion with large impact due to collisions has not been demonstrated on physically simulated character. One exciting future direction is to design new controllers that exploit soft contact to achieve motion with frequent high-speed collisions, such as fighting, parkour, or American football.

A related topic that remains relatively un-explored is the control in the presence of contacts between human characters. Successful rigid body collision algorithms do not lend themselves well to contacts for human motion because human is made of neither passive nor rigid bodies. We believe that the problem of contact with human body is unique and needs to be considered together with the problem of motion control. Our

work takes a step toward designing a better contact model specifically for modeling agile human motion.

REFERENCES

[1] *Autodesk Maya, http://usa.autodesk.com/.*

[2] *Bullet Physics, http://bulletphysics.org/.*

[3] *Havok, http://www.havok.com/.*

[4] *NaturalMotion Endorphin, http://www.naturalmotion.com/endorphin.*

[5] *Open Dynamics Engine (ODE), http://www.ode.org/.*

[6] *PhysX, http://www.nvidia.com/object/physx_new.html.*

[7] Abe, Y., da Silva, M., and Popović, J., "Multiobjective control with frictional contacts," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 249–258, 2007.

[8] Abe, Y. and Popović, J., "Interactive animation of dynamic manipulation," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 195–204, 2006.

[9] Albrecht, I., Haber, J., and Seidel, H.-P., "Construction and animation of anatomically based human hand models," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 98–109, 2003.

[10] Allard, J., Faure, F., Courtecuisse, H., Falipou, F., Duriez, C., and Kry, P. G., "Volume contact constraints at arbitrary resolution," *ACM Trans. Graph.*, vol. 29, 2010.

[11] Allen, B., Curless, B., and Popović, Z., "Articulated body deformation from range scan data," *ACM Transactions on Graphics*, vol. 21, pp. 612–619, July 2002.

[12] Allen, B., Chu, D., Shapiro, A., and Faloutsos, P., "On the beat!: timing and tension for dynamic characters," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 239–247, 2007.

[13] Anitescu, M. and Potra, F. A., "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, pp. 231–247, 1997.

[14] Barbič, J., da Silva, M., and Popović, J., "Deformable object animation using reduced optimal control," *ACM Trans. Graph. (SIGGRAPH)*, vol. 28, no. 3, pp. 1–9, 2009.

[15] Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z., "Interactive skeleton-driven dynamic deformations," *ACM Transactions on Graphics*, vol. 21, pp. 586–593, July 2002.

[16] Chai, J. and Hodgins, J. K., "Constraint-based motion optimization using a statistical dynamic model," *ACM Trans. Graph. (SIGGRAPH)*, vol. 26, no. 3, p. 8, 2007.

[17] Chardonnet, J.-R., David, A., Kheddar, A., and Yokoi, K., "Interactive dynamic simulator for humanoid robotswith deformable soles," in *26th Annual Conference of the Robotics Society of Japan (RSJ)*, 2008.

[18] Cohen, M. F., "Interactive spacetime control for animation," in *SIGGRAPH*, vol. 26, pp. 293–302, July 1992.

[19] Colgate, J. and Schenkel, G., "Passivity of a class of sampled-data systems: Application to haptic interfaces," *Journal of Robotic Systems*, vol. 14, no. 1, pp. 37–47, 1997.

[20] Constantinescu, D., Salcudean, S. E., and Croft, E. A., "Haptic rendering of rigid body collisions," in *International conference on Haptic interfaces for virtual environment and teleoperator systems (HAPTICS)*, pp. 2–8, 2004.

[21] Coros, S., Beaudoin, P., and van de Panne, M., "Generalized biped walking control," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–9, 2010.

[22] Coros, S., Beaudoin, P., Yin, K., and van de Panne, M., "Synthesis of constrained walking skills," *ACM Trans. Graph*, vol. 27, no. 5, p. 113, 2008.

[23] Cutkosky, M. and Kao, I., "Computing and controlling the compliance of a robotic hand," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, 1989.

[24] da Silva, M., Abe, Y., and Popović, J., "Interactive simulation of stylized human locomotion," *ACM Trans. Graph. (SIGGRAPH)*, vol. 27, no. 3, pp. 1–10, 2008.

[25] de Lasa, M. and Hertzmann, A., "Prioritized optimization for task-space control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[26] de Lasa, M., Mordatch, I., and Hertzmann, A., "Feature-based locomotion controllers," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–10, 2010.

[27] Faloutsos, P., van de Panne, M., and Terzopoulos, D., "Dynamic free-form deformations for animation synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 201–214, 1997.

[28] Faloutsos, P., van de Panne, M., and Terzopoulos, D., "Composable controllers for physics-based character animation," in *SIGGRAPH*, pp. 251–260, Aug. 2001.

[29] Fang, A. C. and Pollard, N. S., "Efficient synthesis of physically valid human motion," *ACM Trans. on Graphics (SIGGRAPH)*, pp. 417–426, July 2003.

[30] Farley, C. and Morgenroth, D., "Leg stiffness primarily depends on ankle stiffness during human hopping," *Journal of Biomechanics*, vol. 32, pp. 267–273, 1999.

[31] Galoppo, N., Otaduy, M. A., Tekin, S., Gross, M. H., and Lin, M. C., "Soft articulated characters with fast contact handling," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 243–253, 2007.

[32] Georgopoulos, A., Kalaska, J., and Massey, J., "Spatial trajectories and reaction times of aimed movements: Effects of practice, uncertainty and change in target location," *Journal of Neurophysiology*, vol. 46, pp. 725–743, 1981.

[33] Gill, P., Saunders, M., and Murray, W., "Snopt: An sqp algorithm for large-scale constrained optimization," Tech. Rep. NA 96-2, University of California, San Diego, 1996.

[34] Gourret, J.-P., Thalmann, N. M., and Thalmann, D., "Simulation of object and human skin deformations in a grasping task," in *ACM SIGGRAPH*, pp. 21–30, 1989.

[35] Hauser, K. K., Shen, C., and O'Brien, J. F., "Interactive deformation using modal analysis with constraints," in *Graphics Interface*, pp. 247–256, 2003.

[36] Hodgins, J. K., "Animating human motion," *Scientific American*, vol. 278, pp. 64–69, Mar. 1998.

[37] Hodgins, J. K. and Pollard, N. S., "Adapting simulated behaviors for new characters," pp. 153–162, Aug. 1997.

[38] Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F., "Animating human athletics," in *SIGGRAPH*, pp. 71–78, Aug. 1995.

[39] Isaacs, P. M. and Cohen, M. F., "Controlling dynamic simulation with kinematic constraints," in *SIGGRAPH*, pp. 215–224, 1987.

[40] Jain, S. and Liu, C. K., "Modal-space control for articulated characters," *ACM Trans. Graph.*, vol. 30, 2011.

[41] Jain, S., Ye, Y., and Liu, C. K., "Optimization-based interactive motion synthesis," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1–10, 2009.

[42] James, D. L. and Pai, D. K., "Dyrt: dynamic response textures for real time deformation simulation with graphics hardware," in *SIGGRAPH*, pp. 582–585, 2002.

[43] Kawato, M., "Internal models for motor control and trajectory planning," in *Current Opinions in Neurobiology*, vol. 9, 1999.

[44] Kim, J. and Pollard, N. S., "Direct control of simulated non-human characters," *IEEE Computer Graphics and Applications (In Press)*, 2011.

[45] Kim, J. and Pollard, N. S., "Fast simulation of skeleton-driven deformable body characters," *ACM Transactions on Graphics (In Press)*, 2011.

[46] Kry, P., James, D. L., and Pai, D. K., "Eigenskin: Real time large deformation character skinning in hardware," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2002.

[47] Kry, P. G., Reveret, L., Faure, F., and M.-P.Cani, "Modal locomotion: Animating virtual characters with natural vibrations," *Computer Graphics Forum (Eurographics)*, vol. 28, no. 2, 2009.

[48] Kry, P. G. and Pai, D. K., "Interaction capture and synthesis," *ACM Trans. on Graphics*, vol. 25, pp. 872–880, Aug. 2006.

[49] Kudoh, S., Komura, T., and Ikeuchi, K., "Stepping motion for a human-like character to maintain balance against large perturbations," in *ICRA*, pp. 2661– 2666, May 2006.

[50] Laszlo, J., van de Panne, M., and Fiume, E., "Limit cycle control and its application to the animation of balancing and walking," in *SIGGRAPH*, pp. 155–162, 1996.

[51] Lee, S.-H., Sifakis, E., and Terzopoulos, D., "Comprehensive biomechanical modeling and simulation of the upper body," *ACM Trans. Graph.*, vol. 28, September 2009.

[52] Lee, S.-H. and Terzopoulos, D., "Heads up!: biomechanical modeling and neuromuscular control of the neck," *ACM Trans. Graph. (SIGGRAPH)*, vol. 25, pp. 1188–1198, July 2006.

[53] Lee, Y., Kim, S., and Lee, J., "Data-driven biped control," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–8, 2010.

[54] Lewis, J. P., Cordner, M., and N., F., "Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 19, July 2000.

[55] Liegeois, A., "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[56] Liu, C. K., "Synthesis of interactive hand manipulation," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 163–171, 2008.

[57] Liu, C. K., "Dextrous manipulation from a grasping pose," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 28, no. 3, 2009.

[58] Liu, C. K., Hertzmann, A., and Popović, Z., "Learning physics-based motion style with nonlinear inverse optimization," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 24, pp. 1071–1081, July 2005.

[59] Liu, C. K. and Popović, Z., "Synthesis of complex dynamic character motion from simple animations," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 21, pp. 408–416, July 2002.

[60] Liu, Z., Gortler, S. J., and Cohen, M. F., "Hierarchical spacetime control," in *SIGGRAPH*, pp. 35–42, July 1994.

[61] Lockhart, D. B. and Ting, L. H., "Optimal sensorimotor transformations for balance," *Nat Neurosci*, vol. 10, pp. 1329–1336, Oct. 2007.

[62] Macchietto, A., Zordan, V., and Shelton, C. R., "Momentum control for balance," *ACM Trans. Graph. (SIGGRAPH)*, vol. 28, no. 3, pp. 1–8, 2009.

[63] Maciejewski, A. A. and Klein, C. A., "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Intl. Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.

[64] Magnenat-Thalmann, N., Laperrière, R., and Thalmann, D., "Joint-dependent local deformations for hand animation and object grasping," in *Graphics Interface*, pp. 26–33, June 1988.

[65] Metoyer, R., Zordan, V., Hermens, B., Wu, C.-C., and Soriano, M., "Psychologically inspired anticipation and dynamic response for impacts to the head and upper body," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 173–185, 2008.

[66] Miall, R. C., Weir, D. J., and Stein, J. F., "Visuomotor tracking with delayed visual feedback," *Neuroscience*, vol. 16, no. 3, pp. 511–520, 1985.

[67] Mohr, A. and Gleicher, M., "Building efficient, accurate character skins from examples," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 22, July 2003.

[68] Mordatch, I., de Lasa, M., and Hertzmann, A., "Robust physics-based locomotion using low-dimensional planning," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–8, 2010.

[69] Muico, U., Lee, Y., Popović, J., and Popović, Z., "Contact-aware nonlinear control of dynamic characters," in *ACM Trans. Graph. (SIGGRAPH)*, pp. 1–9, 2009.

[70] Park, S. I. and Hodgins, J. K., "Capturing and animating skin deformation in human motion," *ACM Transactions on Graphics*, vol. 25, pp. 881–889, July 2006.

[71] Pauly, M., Pai, D. K., and Guibas, L. J., "Quasi-rigid objects in contact," in *ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pp. 109–119, 2004.

[72] Popović, Z. and Witkin, A., "Physically based motion transformation," in *SIGGRAPH*, pp. 11–20, Aug. 1999.

[73] Raibert, M. H., *Legged Robots That Balance*. Cambridge, MA, USA: Massachusetts Institute of Technology, 1986.

[74] Ruspini, D. C., Kolarov, K., and Khatib, O., "The haptic display of complex graphical environments," in *SIGGRAPH*, pp. 345–352, 1997.

[75] Safonova, A., Hodgins, J. K., and Pollard, N. S., "Synthesizing physically realistic human motion in low-dimensinal, behavior-specific spaces," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 23, no. 3, pp. 514–521, 2004.

[76] Sentis, L. and Khatib, O., "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[77] Sentis, L. and Khatib, O., "A whole-body control framework for humanoids operating in human environments," in *ICRA*, pp. 2641–2648, May 2006.

[78] Shabana, A. A., *Vibration of Discrete and Continuous Systems*. Springer, 1997.

[79] Sharon, D. and van de Panne, M., "Synthesis of controllers for stylized planar bipedal walking," in *ICRA*, Apr. 2005.

[80] Shiratori, T., Coley, B., Cham, R., and Hodgins, J. K., "Simulating balance recovery responses to trips based on biomechanical principles," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 37–46, 2009.

[81] Sok, K. W., Kim, M., and Lee, J., "Simulating biped behaviors from human motion data," *ACM Trans. Graph. (SIGGRAPH)*, vol. 26, no. 3, p. 107, 2007.

[82] Stewart, A. J. and Cremer, J. F., "Animation of 3d human locomotion: climbing stairs and descending stairs," in *Eurographics Workshop on Animation and Simulation*, pp. 152–168, 1992.

[83] Stewart, A. J. and Cremer, J. F., "Beyond keyframing: an algorithmic approach to animation," in *Graphics Interface*, pp. 273–281, 1992.

[84] Stewart, D. E. and Trinkle, J. C., "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[85] Sulejmanpašić, A. and Popović, J., "Adaptation of performed ballistic motion," *ACM Trans. Graph.*, vol. 24, no. 1, 2005.

[86] Tsang, W., Singh, K., and Eugene, F., "Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 319–328, 2005.

[87] Turner, R. and Thalmann, D., "The elastic surface layer model for animated character construction," in *Proceedings of Computer Graphics International*, Aug. 1993.

[88] Uno, Y., Kawato, M., and Suzuki, R., "Minimum muscle-tension-change model which reproduces human arm movement," in *Symposium on Biological and Physiological Engineering*, pp. 299–302, 1989.

[89] van de Panne, M. and Lamouret, A., "Guided optimization for balanced locomotion," in *Computer Animation and Simulation*, pp. 165–177, Sept. 1995.

[90] Wang, J. M., Fleet, D. J., and Hertzmann, A., "Optimizing walking controllers," *ACM Trans. Graph. (SIGGRAPH Asia)*, vol. 28, no. 5, pp. 1–8, 2009.

[91] Wang, J. M., Fleet, D. J., and Hertzmann, A., "Optimizing walking controllers for uncertain inputs and environments," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–8, 2010.

[92] Wang, X. C. and Phillips, C., "Multi-weight enveloping: Least-squares approximation techniques for skin animation," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2002.

[93] Witkin, A. and Baraff, D., "Physically based modeling: Principles and practice," *SIGGRAPH Course notes*, 1997.

[94] Witkin, A. and Baraff, D., "Physically based modeling," *SIGGRAPH Course notes*, 2001.

[95] Witkin, A. and Kass, M., "Spacetime constraints," in *SIGGRAPH*, vol. 22, pp. 159–168, Aug. 1988.

[96] Wooten, W. L., *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology, 1998.

[97] Wu, J.-c. and Popović, Z., "Terrain-adaptive bipedal locomotion control," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–10, 2010.

[98] Xydas, N. and Kao, I., "Modeling of contact mechanics with experimental results for soft fingers," in *IEEE/RSJ International Conference on Intelligent Robots and System*, 1998.

[99] Yamaguchi, J., Takanishi, A., and Kato, I., "Experimental development of a foot mechanism with shock absorbing material for acquisition of landing surface position information and stabilization of dynamic biped walking," in *IEEE International Conference on Robotics and Automation*, 1995.

[100] Yamane, K. and Nakamura, Y., "Dynamics filter - concept and implementation of on-line motion generator for human figures," in *ICRA*, pp. 688–695, May 2000.

[101] Yamane, K. and Nakamura, Y., "Stable penalty-based model of frictional contacts," in *ICRA*, pp. 1904–1909, May 2006.

[102] Ye, Y. and Liu, C. K., "Animating responsive characters with dynamic constraints in near-unactuated coordinates," *ACM Trans. Graph. (SIGGRAPH Asia)*, vol. 27, no. 5, pp. 1–5, 2008.

[103] Ye, Y. and Liu, C. K., "Optimal feedback control for character animation using an abstract model," *ACM Trans. Graph. (SIGGRAPH)*, vol. 29, no. 4, pp. 1–9, 2010.

[104] Yin, K., Cline, M. B., and Pai, D. K., "Motion perturbation based on simple neuromotor control models," in *Pacific Graphics*, p. 445, 2003.

[105] Yin, K., Loken, K., and van de Panne, M., "Simbicon: simple biped locomotion control," *ACM Trans. Graph. (SIGGRAPH)*, vol. 26, no. 3, p. 105, 2007.

[106] Zilles, C. B. and Salisbury, J. K., "A constraint-based god-object method for haptic display," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 3146–3151, 1995.

[107] Zordan, V., Macchietto, A., Medin, J., Soriano, M., Wu, C.-C., Metoyer, R., and Rose, R., "Anticipation from example," in *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pp. 81–84, 2007.

[108] Zordan, V. B. and Hodgins, J. K., "Motion capture-driven simulations that hit and react," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, pp. 89–96, 2002.

VITA

Sumit Jain was born in the beautiful city of Chandigarh, India in 1983 where he spent most of his childhood. He attended St. Anne's Convent School and D.A.V. College for his secondary and senior secondary education. In the Fall of 2000, Sumit enrolled in the undergraduate program of Computer Science and Engineering at the Indian Institute of Technology (IIT) Delhi located in New Delhi, the capital city of India. During his undergraduate years, Sumit spent the Summer of 2003 in France working as a research intern in the EVASION graphics group at INRIA Rhône-Alpes. He was advised by Dr. Marie-Paule Cani and Dr. François Faure and worked on cloth design and simulation for dressing virtual actors. In his senior year at IIT, Sumit performed research with Dr. Subhashis Banerjee and Dr. Prem Kalra on activity recognition for detecting unusual activities.

After earning his Bachelor of Technology degree, Sumit decided to pursue research in the area of computer vision and graphics and headed over to the University of Southern California (USC) in Los Angeles, California to work with Dr. Ulrich Neumann and Dr. Suya You. He graduated from USC with a Master of Science degree in Computer Science and enrolled in the Ph.D. program at the Georgia Institute of Technology (Georgia Tech) in Atlanta, Georgia in the Fall of 2007. Working with his advisor Dr. C. Karen Liu, he was able to pursue his research interests in the field of physics-based virtual character control and animation. In the Summer of 2011, Sumit completed all the requirements for the Doctoral degree in Computer Science.